Discrete Mathematics

Chapter 2.1

Logical form and logical equivalence

Textbook: Discrete Mathematics with Applications, 4th Ed. - Susanna S. Epp

Why logic?

Logic is the **grammar of mathematics**. It gives precise rules for determining when an argument is valid.

- **Programming**

  Every if/else, while-loop, and boolean expression is propositional logic in action.

- **Digital circuits**

  AND, OR, NOT gates are physical implementations of logical connectives.

- **Database queries**

  SQL WHERE clauses combine conditions with AND, OR, NOT - exactly as in logic.

- **Mathematical proof**

  Every theorem is proved by chaining valid logical arguments.

2.1 Statements (propositions)

A **statement** (or **proposition**) is a sentence that is true or false but not both.
- Epp, Section 2.1

Statements (valid)

"Two plus two equals four."	*True*
"London is the capital of France."	*False*
"1 + 1 = 3"	*False*
"There are infinitely many primes."	*True*

Not statements (invalid)

"He is a college student."	*Depends on who 'he' is*
"x + y > 0"	*Depends on x and y*
"Close the door!"	*Command, not T/F*
"What time is it?"	*Question, not T/F*

2.1 Logical connectives - overview

We use **statement variables** (p, q, r, ...) to represent statements, and **logical connectives** to combine them.

Symbol	Name	Read as	Arabic
$\sim p$	Negation	"not p"	نفي
$p \wedge q$	Conjunction	"p and q"	عطف
$p \vee q$	Disjunction	"p or q"	فصل
$p \oplus q$	Exclusive or	"p or q but not both"	أو حصري

*Each connective has a precise **truth table** that defines its meaning - no ambiguity.*

2.1 Negation (~p)

If *p* is a statement variable, the **negation** of *p* is "not *p*" or "It is not the case that *p*," denoted ~*p*. It has the opposite truth value from *p*.
- Epp, Section 2.1

p	~p
T	F
F	T

Examples

```
p: "It is raining."
~p: "It is not raining."

p: "5 is even."   (F)
~p: "5 is not even."   (T)

p: "2 < 7"   (T)
~p: "2 ≥ 7"   (F)
```

The symbol ~ is called "tilde." Some textbooks use ¬ (hook) or an overbar instead.

2.1 Conjunction ($p \wedge q$)

If *p* and *q* are statement variables, the **conjunction** of *p* and *q* is "*p* and *q*," denoted **p ∧ q**. True when, and only when, both *p* and *q* are true.
- Epp, Section 2.1

p	q	p ∧ q
T	T	T
T	F	F
F	T	F
F	F	F

Important insight

The conjunction is true in exactly one case: both parts must be true. If either part is false, the whole conjunction is false.

Example

`"It is hot and it is sunny."`

True only when BOTH conditions hold.
If it is hot but cloudy, the conjunction is false.

2.1 Disjunction ($p \vee q$)

If *p* and *q* are statement variables, the **disjunction** of *p* and *q* is "*p* or *q*," denoted **p ∨ q**. True when either or both are true; false only when both are false.
- Epp, Section 2.1

p	q	p ∨ q
T	T	T
T	F	T
F	T	T
F	F	F

Inclusive or

In mathematics, "or" always means the inclusive or: p or q or both. The symbol ∨ comes from the Latin word "vel" (inclusive or).

Inclusive vs. exclusive in English

```
"Would you like coffee or tea?"
 – exclusive (pick one)
"Would you like cream or sugar?"
 – inclusive (take both)
```

In logic, ∨ is always inclusive.

2.1 Translating English to symbols

Let h = "It is hot" and s = "It is sunny."

"It is not hot but it is sunny." *"but" means "and"*	$\sim h \wedge s$
"It is neither hot nor sunny." *"neither...nor" = "not...and not..."*	$\sim h \wedge \sim s$
"It is hot or it is not sunny." *standard or with negation*	$h \vee \sim s$

Inequalities and logic

$a \leq x \leq b$ means $a \leq x$ **and** $x \leq b$

$x \leq a$ means $x < a$ **or** $x = a$

Epp, Examples 2.1.2 - 2.1.3

2.1 Exclusive or ($p \oplus q$)

The **exclusive or** is true when exactly one of p, q is true - not both. It is defined as:

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

p	q	$p \vee q$	$p \wedge q$	$\sim(p \wedge q)$	$(p \vee q) \wedge \sim(p \wedge q)$
T	T	T	T	F	F
T	F	T	F	T	T
F	T	T	F	T	T
F	F	F	F	T	F

True when exactly one is true. False when both are true or both are false.

Epp, Example 2.1.4

2.1 Building truth tables

A truth table systematically lists all possible combinations of truth values.

1. **Count variables**
   n variables give 2^n rows (2 vars = 4 rows, 3 vars = 8 rows).
2. **List all combinations**
   Standard order: start all T, toggle rightmost column fastest.
3. **Add columns for sub-expressions**
   Build up: compute innermost parts first, then combine.
4. **Fill in using definitions**
   Apply each connective's truth table to compute values.

Standard row ordering for two variables:

Row	p	q
1	T	T
2	T	F
3	F	T
4	F	F

Evaluation order: ~ first, then ∧, then ∨ (like PEMDAS for logic).
"Parentheses, Exponents, Multiplication, Division, Addition, Subtraction"

2.1 Example: $(p \wedge q) \vee \sim r$

Three variables = 2^3 = 8 rows. Build step by step.

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

Since $(p \wedge q) \vee \sim r$ is an OR statement, it is false only when both parts are false.

Epp, Example 2.1.5

2.1 Logical equivalence

Two statement forms are called **logically equivalent** if, and only if, they have identical truth values for each possible substitution of statements for their statement variables. The logical equivalence of *P* and *Q* is denoted **P ≡ Q**.

- Epp, Section 2.1

Example: $p \wedge q \equiv q \wedge p$ (commutative law)

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

$p \wedge q$ and $q \wedge p$ always have the same truth values, so they are logically equivalent.

2.1 Equivalence and non-equivalence

Double negation: $\sim(\sim p) \equiv p$

p	~p	~(~p)
T	F	T
F	T	F

Same truth values - equivalent.

Non-equivalence: $\sim(p \wedge q) \not\equiv \sim p \wedge \sim q$

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

Rows 2 and 3 differ - NOT equivalent.
Do not "distribute" negation over ∧!

Common mistake: to assume $\sim(p \wedge q) = \sim p \wedge \sim q$. The truth table proves this is wrong.

2.1 De Morgan's laws

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

Negate an AND -> becomes OR with each part negated.

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

Negate an OR -> becomes AND with each part negated.

Proof of the first law:

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

$\sim(p \wedge q)$ and $\sim p \vee \sim q$ always have the same truth values, confirming the equivalence.

2.1 Applying De Morgan's laws

Statement: "John is 6 feet tall and he weighs at least 200 pounds."

Negation: "John is not 6 feet tall or he weighs less than 200 pounds."

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

Statement: "The bus was late or Ali’s watch was slow."

Negation: "The bus was not late and Ali's watch was not slow."

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

Statement: $-2 < x < 7$ (means $-2 < x$ AND $x < 7$)

Negation: $x \leq -2$ OR $x \geq 7$

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

Remember: when negating an inequality, $\sim(x < a)$ becomes $x \geq a$, not $x > a$.

Epp, Example 2.1.9

2.1 Tautologies and contradictions

A **tautology** is a statement form that is always true regardless of the truth values of its variables. A **contradiction** is a statement form that is always false.
- Epp, Section 2.1

$p \vee \sim p$ is a tautology (denoted t)

p	~p	$p \vee \sim p$
T	F	T
F	T	T

All T's - always true.

$p \wedge \sim p$ is a contradiction (denoted c)

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

All F's - always false.

Identity laws: $p \wedge t \equiv p$ $\quad p \vee c \equiv p$ $\quad p \vee t \equiv t$ $\quad p \wedge c \equiv c$

A tautology is the "1" and a contradiction is the "0" of logic.

2.1 The distributive law

Just as in algebra a(b + c) = ab + ac, logic has a distributive law. It works in **both directions**: expanding (left to right) and factoring (right to left).
- Epp, Theorem 2.1.1, Law 3

Expanding (left → right)

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

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

Distribute p across each term inside the parentheses.

Factoring (right → left)

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

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

Spot the common factor p, pull it out. This is the form used in proofs.

Example of factoring:

$(p \vee \sim q) \wedge (p \vee q)$ common factor is p, leftovers are ~q and q $\equiv$ $p \vee (\sim q \wedge q)$

Epp, Theorem 2.1.1, Law 3

2.1 Summary of logical equivalences

Theorem 2.1.1 (Epp, p. 35)

Law	Conjunction form	Disjunction form
1. Commutative	$p \wedge q \equiv q \wedge p$	$p \vee q \equiv q \vee p$
2. Associative	$(p \wedge q) \wedge r \equiv p \wedge (q \wedge r)$	$(p \vee q) \vee r \equiv p \vee (q \vee r)$
3. Distributive	$p \wedge (q \vee r) \equiv (p \wedge q) \vee (p \wedge r)$	$p \vee (q \wedge r) \equiv (p \vee q) \wedge (p \vee r)$
4. Identity	$p \wedge t \equiv p$	$p \vee c \equiv p$
5. Negation	$p \vee {\sim}p \equiv t$	$p \wedge {\sim}p \equiv c$
6. Double negative	${\sim}({\sim}p) \equiv p$	
7. Idempotent	$p \wedge p \equiv p$	$p \vee p \equiv p$
8. Universal bound	$p \vee t \equiv t$	$p \wedge c \equiv c$
9. De Morgan's	${\sim}(p \wedge q) \equiv {\sim}p \vee {\sim}q$	${\sim}(p \vee q) \equiv {\sim}p \wedge {\sim}q$
10. Absorption	$p \vee (p \wedge q) \equiv p$	$p \wedge (p \vee q) \equiv p$
11. Negations of t, c	${\sim}t \equiv c$	${\sim}c \equiv t$

2.1 Simplifying statement forms

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

$\sim(\sim p \wedge q) \wedge (p \vee q)$	$\equiv (\sim(\sim p) \vee \sim q) \wedge (p \vee q)$	*by De Morgan's laws*
	$\equiv (p \vee \sim q) \wedge (p \vee q)$	*by the double negative law*
	$\equiv p \vee (\sim q \wedge q)$	*by the distributive law*
	$\equiv p \vee (q \wedge \sim q)$	*by the commutative law for* $\wedge$
	$\equiv p \vee c$	*by the negation law*
	$\equiv \mathbf{p}$	*by the identity law*

Each step replaces a sub-expression with a logically equivalent one from Theorem 2.1.1.

2.1 Logic in code

Every logical connective has a direct programming equivalent.

Logic	Python	Java / C	SQL
$\sim p$	not p	!p	NOT p
$p \wedge q$	p and q	p && q	p AND q
$p \vee q$	p or q	p || q	p OR q
$p \oplus q$	p ^ q	p ^ q	p XOR q (MySQL)

De Morgan's laws in Python

```
not (is_admin and is_active)          # ~(p ∧ q)
(not is_admin) or (not is_active)     # ~p ∨ ~q

These two expressions always return the same result - De Morgan's law!
```

Summary

- **A statement is either true or false**

  Questions, commands, and expressions with free variables are not statements.

- **Connectives combine statements precisely**

  ~ (not), ∧ (and), ∨ (or), ⊕ (xor) each have a fixed truth table.

- **Truth tables enumerate all possibilities**

  n variables produce 2^n rows. Build step by step through intermediate columns.

- **Logical equivalence means identical truth values**

  $P \equiv Q$ if every row of their truth tables matches.

- **De Morgan's laws negate compound statements**

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

- **Theorem 2.1.1 is the toolbox for simplification**

  Use the 11 equivalence laws to reduce complex expressions.

Next session

Chapter 2.2

Conditional statements

If p then q ($p \to q$)

The truth table for "if-then" - why a false hypothesis makes the conditional true

Converse, inverse, and contrapositive

Biconditional: p if and only if q ($p \leftrightarrow q$)