# Mathematical Logic

Part One

***Question:*** How do we formalize the definitions and reasoning we use in our proofs?

# Where We're Going

- ***Propositional Logic*** (Today)
  - Basic logical connectives.
  - Truth tables.
  - Logical equivalences.
- ***First-Order Logic*** (Wednesday/Friday)
  - Reasoning about properties of multiple objects.

# Propositional Logic

A ***proposition*** is a statement that is, by itself, either true or false.

# Some Sample Propositions

- Puppies are cuter than kittens.
- Kittens are cuter than puppies.
- Usain Bolt can outrun everyone in this room.
- CS103 is useful for cocktail parties.
- This is the last entry on this list.

# More Propositions

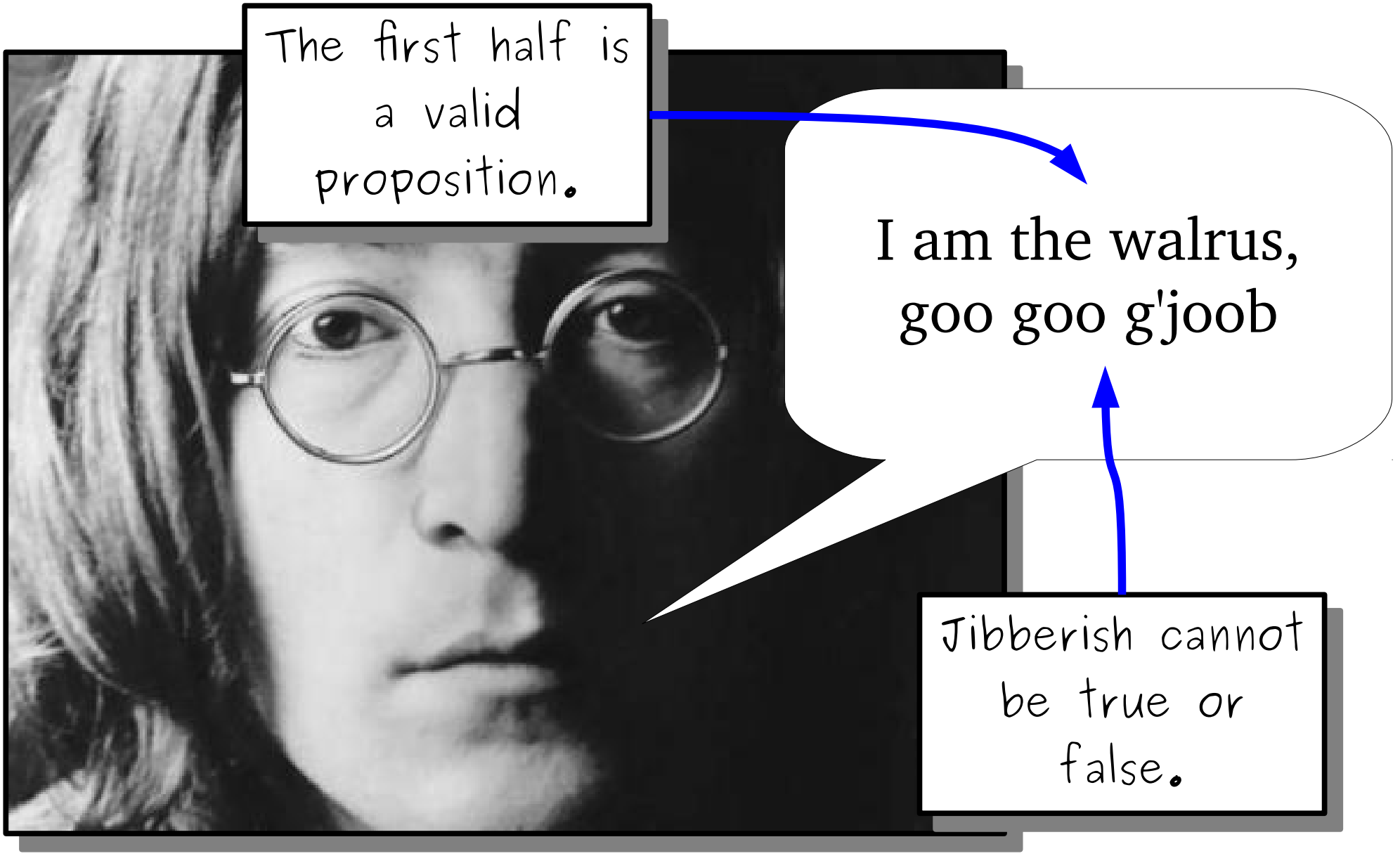
- They say time's supposed to heal ya.
- But I ain't done much healing.
- I'm in California dreaming about who we used to be.
- I've forgotten how it felt before the world fell at our feet.
- There's such a difference between us.

# Things That Aren't Propositions

# Things That Aren't Propositions


Questions cannot be true or false.

# Things That Aren't Propositions



# Propositional Logic

- ***Propositional logic*** is a mathematical system for reasoning about propositions and how they relate to one another.
- Every statement in propositional logic consists of ***propositional variables*** combined via ***propositional connectives***.
  - Each variable represents some proposition, such as “You liked it” or “You should have put a ring on it.”
  - Connectives encode how propositions are related, such as “If you liked it, then you should have put a ring on it.”

# Propositional Variables

- Each proposition will be represented by a ***propositional variable***.
- Propositional variables are usually represented as lower-case letters, such as $p$, $q$, $r$, $s$, etc.
- Each variable can take one one of two values: true or false.

# Propositional Connectives

- **Logical NOT: $\neg p$**
  - Read "***not*** $p$"
  - $\neg p$ is true if and only if p is false.
  - Also called ***logical negation***.
- **Logical AND: $p \wedge q$**
  - Read "$p$ ***and*** $q$."
  - $p \wedge q$ is true if both p and q are true.
  - Also called ***logical conjunction***.
- **Logical OR: $p \vee q$**
  - Read "$p$ ***or*** $q$."
  - $p \vee q$ is true if at least one of $p$ or $q$ are true (inclusive OR)
  - Also called ***logical disjunction***.

# Truth Tables

- A ***truth table*** is a table showing the truth value of a propositional logic formula as a function of its inputs.
- Useful for several reasons:
  - They give a formal definition of what a connective “means.”
  - They give us a way to figure out what a complex propositional formula says.

# The Truth Table Tool

# Summary of Important Points

- The ∨ connective is an *inclusive* "or." It's true if at least one of the operands is true.
  - Similar to the || operator in C, C++, Java and the `or` operator in Python.
- If we need an exclusive "or" operator, we can build it out of what we already have.

# Mathematical Implication

# Implication

- The → connective is used to represent implications.
  - Its technical name is the ***material conditional*** operator.
- What is its truth table?

# Why This Truth Table?

- The truth values of the → are the way they are because they're *defined* that way.
- The intuition:
  - We want $p \to q$ to mean "if $p$ is true, $q$ is true as well."
  - The only way this *doesn't* happen is if $p$ is true and $q$ is false.
  - In other words, $p \to q$ should be true whenever $\neg(p \wedge \neg q)$ is true.
  - What's the truth table for $\neg(p \wedge \neg q)$?

# Truth Table for Implication

| $p$ | $q$ | $p \rightarrow q$ |
|---|---|---|
| F | F | T |
| F | T | T |
| T | F | F |
| T | T | T |

The only way for p → q to be false is for p to be true and q to be false. Otherwise, p → q is by definition true.

# The Biconditional Connective

# The Biconditional Connective

- The biconditional connective $\leftrightarrow$ is used to represent a two-directional implication.
- Specifically, $p \leftrightarrow q$ means that $p$ implies $q$ and $q$ implies $p$.
- What should its truth table look like?

# Biconditionals

- The ***biconditional*** connective $p \leftrightarrow q$ is read "$p$ if and only if $q$."
- Here's its truth table:

| $p$ | $q$ | $p \leftrightarrow q$ |
|---|---|---|
| F | F | T |
| F | T | F |
| T | F | F |
| T | T | T |

One interpretation of ↔ is to think of it as equality: the two propositions must have equal truth values.

# True and False

- There are two more “connectives” to speak of: true and false.
  - The symbol $\top$ is a value that is always true.
  - The symbol $\bot$ is value that is always false.
- These are often called connectives, though they don't connect anything.
  - (Or rather, they connect zero things.)

# Proof by Contradiction

- Suppose you want to prove $p$ is true using a proof by contradiction.
- The setup looks like this:
  - Assume $p$ is false.
  - Derive something that we know is false.
  - Conclude that $p$ is true.
- In propositional logic:

$$\mathbf{(\neg p \to \bot) \to p}$$

# Operator Precedence

- How do we parse this statement?

$$\neg x \to y \vee z \to x \vee y \wedge z$$

- Operator precedence for propositional logic:

$\neg$

$\wedge$

$\vee$

$\to$

$\leftrightarrow$

- All operators are right-associative.
- We can use parentheses to disambiguate.

# Operator Precedence

- How do we parse this statement?

$$\neg x \to y \vee z \to x \vee y \wedge z$$

- Operator precedence for propositional logic:

$\neg$

$\wedge$

$\vee$

$\to$

$\leftrightarrow$

- All operators are right-associative.
- We can use parentheses to disambiguate.

# Operator Precedence

- How do we parse this statement?

$$(\neg x) \to y \vee z \to x \vee y \wedge z$$

- Operator precedence for propositional logic:

$\neg$

$\wedge$

$\vee$

$\to$

$\leftrightarrow$

- All operators are right-associative.
- We can use parentheses to disambiguate.

# Operator Precedence

- How do we parse this statement?

$$(\neg x) \to y \vee z \to x \vee y \wedge z$$

- Operator precedence for propositional logic:

¬

∧

∨

→

↔

- All operators are right-associative.
- We can use parentheses to disambiguate.

# Operator Precedence

- How do we parse this statement?

$$(\neg x) \to y \vee z \to x \vee (y \wedge z)$$

- Operator precedence for propositional logic:

$\neg$

$\wedge$

$\vee$

$\to$

$\leftrightarrow$

- All operators are right-associative.
- We can use parentheses to disambiguate.

# Operator Precedence

- How do we parse this statement?

$$(\neg x) \rightarrow y \vee z \rightarrow x \vee (y \wedge z)$$

- Operator precedence for propositional logic:

$\neg$

$\wedge$

$\vee$

$\rightarrow$

$\leftrightarrow$

- All operators are right-associative.
- We can use parentheses to disambiguate.

# Operator Precedence

- How do we parse this statement?

$$(\neg x) \rightarrow (y \vee z) \rightarrow (x \vee (y \wedge z))$$

- Operator precedence for propositional logic:

$\neg$

$\wedge$

$\vee$

$\rightarrow$

$\leftrightarrow$

- All operators are right-associative.
- We can use parentheses to disambiguate.

# Operator Precedence

- How do we parse this statement?

$$(\neg x) \rightarrow (y \vee z) \rightarrow (x \vee (y \wedge z))$$

- Operator precedence for propositional logic:

$\neg$

$\wedge$

$\vee$

$\rightarrow$

$\leftrightarrow$

- All operators are right-associative.
- We can use parentheses to disambiguate.

# Operator Precedence

- How do we parse this statement?

$$(\neg x) \rightarrow ((y \vee z) \rightarrow (x \vee (y \wedge z)))$$

- Operator precedence for propositional logic:

$\neg$

$\wedge$

$\vee$

$\rightarrow$

$\leftrightarrow$

- All operators are right-associative.
- We can use parentheses to disambiguate.

# Operator Precedence

- How do we parse this statement?

$$(\neg x) \rightarrow ((y \vee z) \rightarrow (x \vee (y \wedge z)))$$

- Operator precedence for propositional logic:

$\neg$

$\wedge$

$\vee$

$\rightarrow$

$\leftrightarrow$

- All operators are right-associative.
- We can use parentheses to disambiguate.

# Operator Precedence

- The main points to remember:
  - ¬ binds to whatever immediately follows it.
  - ∧ and ∨ bind more tightly than →.
- We will commonly write expressions like $p \wedge q \rightarrow r$ without adding parentheses.
- For more complex expressions, we'll try to add parentheses.
- Confused? Just ask!

Time-Out for Announcements!

# Problem Set One

- The checkpoint problem for PS1 was due at the start of class today.
  - We'll try to have it graded and returned by the start of Wednesday's class.
- The remaining problems from PS1 are due on Friday.
  - Have questions? Stop by office hours, or ask on Piazza, or email the staff list!

# Solution Sets

- We've released hardcopy solutions to the checkpoint problem for Problem Set One.
- ***Please be sure to read the solution sets in this course.*** The solution sets give sample solutions, which can be super useful for learning how to write proofs. For the full problem sets, we also explain why we asked each question and what insights we were hoping you would have.
- ***Solutions are not available electronically.*** We release solutions in hardcopy only. If you didn't pick them up in class, you can pick them up in the Gates building. (I'll show you where in a second).
  - SCPD students: we'll send solutions to your site monitors, who will then distribute them to you.

# Where to Get Solutions


STANFORD ENGINEERING
VENTURE FUND LABORATORIES
WILLIAM GATES
COMPUTER SCIENCE


CS106 STAFF
CS106A
CS106B
Lecturer TA
CS106 Coordinators
Section Leaders
Course Helpers
CS106A
CS106B
CS106X
EE106


CS 106A (1/4)
CS 106A (2/4)
CS 109
CS 106A (3/4)
CS 106A (4/4)
CS103 Handouts
CS243
Handouts will be available here...
CS103 Handouts
... and here, if we run out of space. ☺

# Your Questions

# “Why do things like computability and complexity theory matter to a Software Developer?”

Many fundamental aspects of computers and programming (the need to debug things, multithreading, virtualization, optimization, etc.) are all fundamentally motivated by deep results from computability theory. If we have time later this quarter, we'll see why, for example, debugging is necessary, recursion is inescapable and virtualization must be possible.

Complexity theory helps us better understand why certain key problems are hard to solve and what to do when we find them. You'll almost certainly encounter problems whose hardness is theoretically motivated in the course of a software career. If you don't, aim higher! ☺

“Can parallelism impact tractability and/or computability?”

The answer is either “it depends” or “nobody knows.” We'll explore this later in the quarter.

Back to CS103!

# Recap So Far

- A ***propositional variable*** is a variable that is either true or false.
- The ***propositional connectives*** are
  - Negation: $\neg p$
  - Conjunction: $p \wedge q$
  - Disjunction: $p \vee q$
  - Implication: $p \rightarrow q$
  - Biconditional: $p \leftrightarrow q$
  - True: $\top$
  - False: $\bot$

# Translating into Propositional Logic

# Some Sample Propositions

$a$: I will be awake this evening.

$b$: There is a lunar eclipse this evening.

$c$: I will see the lunar eclipse.

# Some Sample Propositions

$a$: I will be awake this evening.

$b$: There is a lunar eclipse this evening.

$c$: I will see the lunar eclipse.

"I won't see a lunar eclipse if I'm not awake this evening."

# Some Sample Propositions

$a$: I will be awake this evening.

$b$: There is a lunar eclipse this evening.

$c$: I will see the lunar eclipse.

"I won't see a lunar eclipse if I'm not awake this evening."

$$\neg a \rightarrow \neg c$$

$$\text{“}p \text{ if } q\text{”}$$

translates to

$$q \rightarrow p$$

It does *not* translate to

$$p \rightarrow q$$

# Some Sample Propositions

*a*: I will be awake this evening.

*b*: There is a lunar eclipse this evening.

*c*: I will see a lunar eclipse.

# Some Sample Propositions

*a*: I will be awake this evening.

*b*: There is a lunar eclipse this evening.

*c*: I will see a lunar eclipse.

"If I will be awake this evening, but there's no lunar eclipse, I won't see a lunar eclipse.

# Some Sample Propositions

$a$: I will be awake this evening.

$b$: There is a lunar eclipse this evening.

$c$: I will see a lunar eclipse.

"If I will be awake this evening, but there's no lunar eclipse, I won't see a lunar eclipse.

$$a \wedge \neg b \rightarrow \neg c$$

“$p$, but $q$”

translates to

$$p \wedge q$$

# The Takeaway Point

- When translating into or out of propositional logic, be very careful not to get tripped up by nuances of the English language.
  - In fact, this is one of the reasons we have a symbolic notation in the first place!
- Many prepositional phrases lead to counterintuitive translations; make sure to double-check yourself!

# Propositional Equivalences

***Quick Question:***

What would I have to show you to convince you that the statement $p \wedge q$ is false?

***Quick Question:***

What would I have to show you to convince you that the statement $p \vee q$ is false?

# De Morgan's Laws

- Using truth tables, we concluded that

$$\neg(p \wedge q)$$

  is equivalent to

$$\neg p \vee \neg q$$

- We also saw that

$$\neg(p \vee q)$$

  is equivalent to

$$\neg p \wedge \neg q$$

- These two equivalences are called ***De Morgan's Laws***.

# De Morgan's Laws in Code

- ***Pro tip:*** Don't write this:

```
if (!(p() && q()) {
    /* … */
}
```

- Write this instead:

```
if (!p() || !q()) {
    /* … */
}
```

- (This even short-circuits correctly!)

# Logical Equivalence

- Because $\neg(p \wedge q)$ and $\neg p \vee \neg q$ have the same truth tables, we say that they're ***equivalent*** to one another.
- We denote this by writing

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

- The $\equiv$ symbol is not a connective.
  - The statement $\neg(p \wedge q) \leftrightarrow (\neg p \vee \neg q)$ is a propositional formula. If you plug in different values of $p$ and $q$, it will evaluate to a truth value. It just happens to evaluate to true every time.
  - The statement $\neg(p \wedge q) \equiv \neg p \vee \neg q$ means "these two formulas have exactly the same truth table."
- In other words, the notation $\varphi \equiv \psi$ means "$\varphi$ and $\psi$ always have the same truth values, regardless of how the variables are assigned."

# An Important Equivalence

- Earlier, we talked about the truth table for $p \to q$. We chose it so that

$$\boldsymbol{p \to q \quad \equiv \quad \neg(p \wedge \neg q)}$$

- Later on, this equivalence will be incredibly useful:

$$\boldsymbol{\neg(p \to q) \quad \equiv \quad p \wedge \neg q}$$

# Another Important Equivalence

- Here's a useful equivalence. Start with

$$p \to q \equiv \neg(p \wedge \neg q)$$

- By De Morgan's laws:

$$\begin{aligned} p \to q &\equiv \neg(p \wedge \neg q) \\ &\equiv \neg p \vee \neg\neg q \\ &\equiv \neg p \vee q \end{aligned}$$

- Thus $p \to q \equiv \neg p \vee q$

# Another Important Equivalence

- Here's a useful equivalence. Start with

$$p \to q \equiv \neg(p \wedge \neg q)$$

- By De Morgan's laws:

$$p \to q \equiv \neg(p \wedge \neg q)$$

$$\equiv \neg p \vee \neg\neg q$$

$$\equiv \neg p \vee$$

- Thus $p \to q \equiv \neg p \vee q$

If $p$ is false, then $\neg p \vee q$ is true. If $p$ is true, then $q$ has to be true for the whole expression to be true.

# One Last Equivalence

# The Contrapositive

- The contrapositive of the statement

$$p \rightarrow q$$

  is the statement

$$\neg q \rightarrow \neg p$$

- These are logically equivalent, which is why proof by contrapositive works:

$$\boldsymbol{p \rightarrow q \quad \equiv \quad \neg q \rightarrow \neg p}$$

# Why All This Matters

# Why All This Matters

- Suppose we want to prove the following statement:

  "If $x + y = 16$, then $x \geq 8$ or $y \geq 8$"

# Why All This Matters

- Suppose we want to prove the following statement:

"If $x + y = 16$, then $x \geq 8$ or $y \geq 8$"

$$x + y = 16 \rightarrow x \geq 8 \vee y \geq 8$$

# Why All This Matters

- Suppose we want to prove the following statement:

"If $x + y = 16$, then $x \geq 8$ or $y \geq 8$"

$$x + y = 16 \rightarrow x \geq 8 \vee y \geq 8$$

# Why All This Matters

- Suppose we want to prove the following statement:

  "If $x + y = 16$, then $x \geq 8$ or $y \geq 8$"

  $$\neg(x \geq 8 \vee y \geq 8) \rightarrow \neg(x + y = 16)$$

# Why All This Matters

- Suppose we want to prove the following statement:

"If $x + y = 16$, then $x \geq 8$ or $y \geq 8$"

$$\neg(x \geq 8 \vee y \geq 8) \rightarrow \neg(x + y = 16)$$

# Why All This Matters

- Suppose we want to prove the following statement:

"If $x + y = 16$, then $x \geq 8$ or $y \geq 8$"

$\neg(x \geq 8 \vee y \geq 8) \rightarrow \neg(x + y = 16)$

# Why All This Matters

- Suppose we want to prove the following statement:

"If $x + y = 16$, then $x \geq 8$ or $y \geq 8$"

$\neg(x \geq 8 \vee y \geq 8) \rightarrow x + y \neq 16$

# Why All This Matters

- Suppose we want to prove the following statement:

"If $x + y = 16$, then $x \geq 8$ or $y \geq 8$"

$$\neg(x \geq 8 \vee y \geq 8) \rightarrow x + y \neq 16$$

# Why All This Matters

- Suppose we want to prove the following statement:

"If $x + y = 16$, then $x \geq 8$ or $y \geq 8$"

$$\neg(x \geq 8 \vee y \geq 8) \rightarrow x + y \neq 16$$

# Why All This Matters

- Suppose we want to prove the following statement:

"If $x + y = 16$, then $x \geq 8$ or $y \geq 8$"

$$\neg(x \geq 8) \wedge \neg(y \geq 8) \rightarrow x + y \neq 16$$

# Why All This Matters

- Suppose we want to prove the following statement:

"If $x + y = 16$, then $x \geq 8$ or $y \geq 8$"

$$\neg(x \geq 8) \wedge \neg(y \geq 8) \rightarrow x + y \neq 16$$

# Why All This Matters

- Suppose we want to prove the following statement:

"If $x + y = 16$, then $x \geq 8$ or $y \geq 8$"

$\neg(x \geq 8) \wedge \neg(y \geq 8) \rightarrow x + y \neq 16$

# Why All This Matters

- Suppose we want to prove the following statement:

"If $x + y = 16$, then $x \geq 8$ or $y \geq 8$"

$x < 8 \wedge \neg(y \geq 8) \rightarrow x + y \neq 16$

# Why All This Matters

- Suppose we want to prove the following statement:

"If $x + y = 16$, then $x \geq 8$ or $y \geq 8$"

$$x < 8 \land \neg(y \geq 8) \rightarrow x + y \neq 16$$

# Why All This Matters

- Suppose we want to prove the following statement:

"If $x + y = 16$, then $x \geq 8$ or $y \geq 8$"

$$x < 8 \wedge \neg(y \geq 8) \rightarrow x + y \neq 16$$

# Why All This Matters

- Suppose we want to prove the following statement:

"If $x + y = 16$, then $x \geq 8$ or $y \geq 8$"

$$x < 8 \wedge y < 8 \rightarrow x + y \neq 16$$

# Why All This Matters

- Suppose we want to prove the following statement:

"If $x + y = 16$, then $x \geq 8$ or $y \geq 8$"

$$x < 8 \wedge y < 8 \rightarrow x + y \neq 16$$

# Why All This Matters

- Suppose we want to prove the following statement:

"If $x + y = 16$, then $x \geq 8$ or $y \geq 8$"

$$x < 8 \wedge y < 8 \rightarrow x + y \neq 16$$

"If $x < 8$ and $y < 8$, then $x + y \neq 16$"

***Theorem:*** If $x + y = 16$, then $x \geq 8$ or $y \geq 8$.

***Proof:*** By contrapositive. We will prove that if $x < 8$ and $y < 8$, then $x + y \neq 16$. To see this, note that

$$\begin{aligned} x + y &< 8 + y \\ &< 8 + 8 \\ &= 16 \end{aligned}$$

This means that $x + y < 16$, so $x + y \neq 16$, which is what we needed to show. ■

# Why All This Matters

- Suppose we want to prove the following statement:

  "If $x + y = 16$, then $x \geq 8$ or $y \geq 8$"

# Why All This Matters

- Suppose we want to prove the following statement:

"If $x + y = 16$, then $x \geq 8$ or $y \geq 8$"

$$x + y = 16 \rightarrow x \geq 8 \vee y \geq 8$$

# Why All This Matters

- Suppose we want to prove the following statement:

"If $x + y = 16$, then $x \geq 8$ or $y \geq 8$"

$$\neg(x + y = 16 \rightarrow x \geq 8 \vee y \geq 8)$$

# Why All This Matters

- Suppose we want to prove the following statement:

"If $x + y = 16$, then $x \geq 8$ or $y \geq 8$"

$$\neg(x + y = 16 \rightarrow x \geq 8 \vee y \geq 8)$$

# Why All This Matters

- Suppose we want to prove the following statement:

"If $x + y = 16$, then $x \geq 8$ or $y \geq 8$"

$$x + y = 16 \wedge \neg(x \geq 8 \vee y \geq 8)$$

# Why All This Matters

- Suppose we want to prove the following statement:

"If $x + y = 16$, then $x \geq 8$ or $y \geq 8$"

$$x + y = 16 \wedge \neg(x \geq 8 \vee y \geq 8)$$

# Why All This Matters

- Suppose we want to prove the following statement:

"If $x + y = 16$, then $x \geq 8$ or $y \geq 8$"

$$x + y = 16 \wedge \neg(x \geq 8 \vee y \geq 8)$$

# Why All This Matters

- Suppose we want to prove the following statement:

"If $x + y = 16$, then $x \geq 8$ or $y \geq 8$"

$$x + y = 16 \wedge \neg(x \geq 8) \wedge \neg(y \geq 8)$$

# Why All This Matters

- Suppose we want to prove the following statement:

"If $x + y = 16$, then $x \geq 8$ or $y \geq 8$"

$$x + y = 16 \wedge \neg(x \geq 8) \wedge \neg(y \geq 8)$$

# Why All This Matters

- Suppose we want to prove the following statement:

"If $x + y = 16$, then $x \geq 8$ or $y \geq 8$"

$$x + y = 16 \wedge \neg(x \geq 8) \wedge \neg(y \geq 8)$$

# Why All This Matters

- Suppose we want to prove the following statement:

"If $x + y = 16$, then $x \geq 8$ or $y \geq 8$"

$$x + y = 16 \wedge x < 8 \wedge \neg(y \geq 8)$$

# Why All This Matters

- Suppose we want to prove the following statement:

"If $x + y = 16$, then $x \geq 8$ or $y \geq 8$"

$$x + y = 16 \wedge x < 8 \wedge \neg(y \geq 8)$$

# Why All This Matters

- Suppose we want to prove the following statement:

"If $x + y = 16$, then $x \geq 8$ or $y \geq 8$"

$$x + y = 16 \wedge x < 8 \wedge \neg(y \geq 8)$$

# Why All This Matters

- Suppose we want to prove the following statement:

"If $x + y = 16$, then $x \geq 8$ or $y \geq 8$"

$$x + y = 16 \wedge x < 8 \wedge y < 8$$

# Why All This Matters

- Suppose we want to prove the following statement:

"If $x + y = 16$, then $x \geq 8$ or $y \geq 8$"

$$x + y = 16 \wedge x < 8 \wedge y < 8$$

# Why All This Matters

- Suppose we want to prove the following statement:

"If $x + y = 16$, then $x \geq 8$ or $y \geq 8$"

$$x + y = 16 \wedge x < 8 \wedge y < 8$$

"$x + y = 16$, but $x < 8$ and $y < 8$."

***Theorem:*** If $x + y = 16$, then $x \geq 8$ or $y \geq 8$.

***Proof:*** Assume for the sake of contradiction that $x + y = 16$, but that $x < 8$ and $y < 8$. Then

$$\begin{aligned} x + y &< 8 + y \\ &< 8 + 8 \\ &= 16 \end{aligned}$$

So $x + y < 16$, contradicting that $x + y = 16$. We have reached a contradiction, so our assumption must have been wrong. Therefore if $x + y = 16$, then $x \geq 8$ or $y \geq 8$. ■

# Why This Matters

- Propositional logic is a tool for reasoning about how various statements affect one another.
- To better understand how to prove a result, it often helps to translate what you're trying to prove into propositional logic first.
- That said, propositional logic isn't expressive enough to capture all statements. For that, we need something more powerful.