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*** (Friday/Monday)
  - Reasoning about properties of multiple objects.

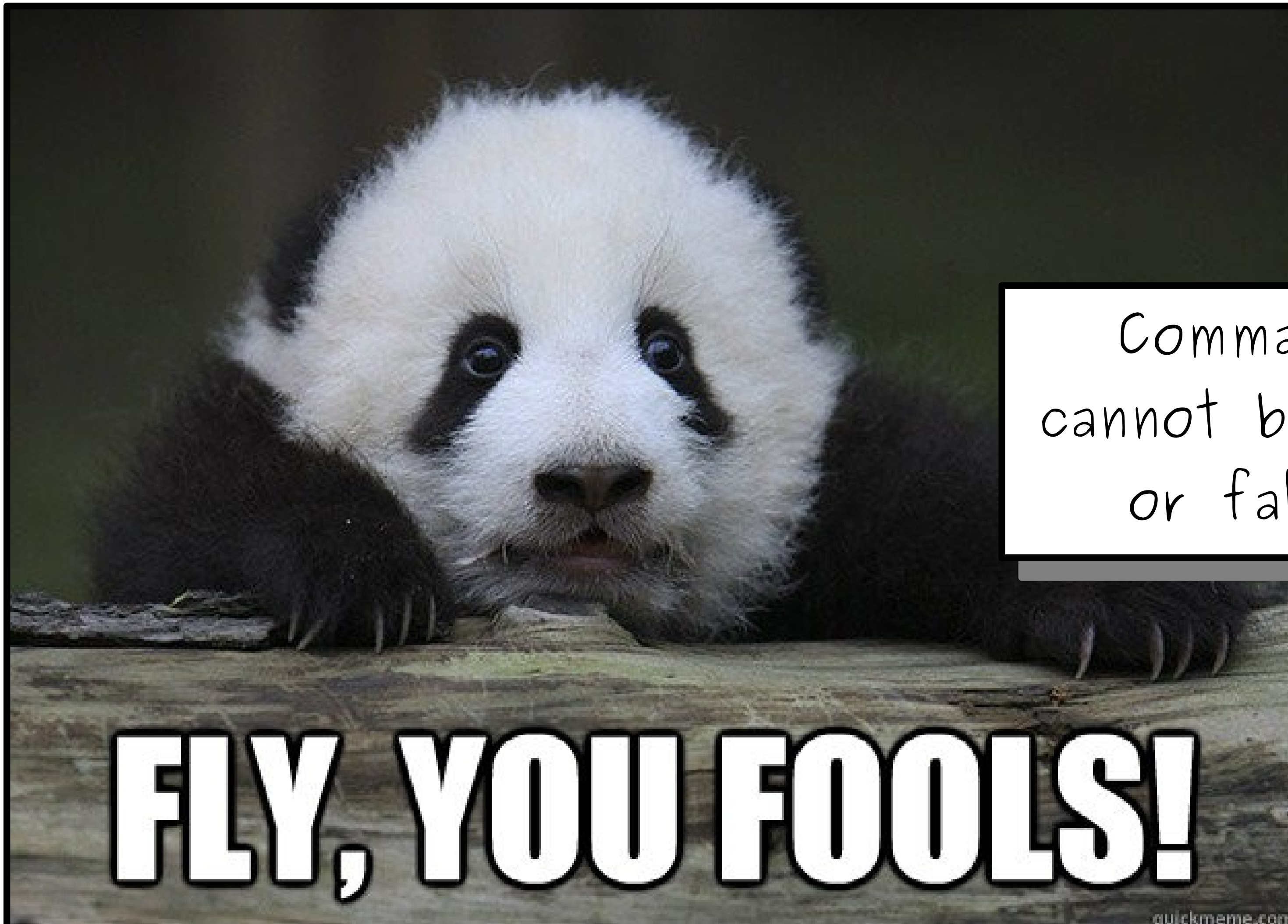
Propositional Logic

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.

Things That Aren't Propositions



Commands cannot be true or false.

Things That Aren't Propositions

Questions cannot be true or false.

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 and only 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 and only 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.

Truth Table for XOR

This is the truth table for XOR. ***You choose*** how we can write XOR using the other logical operators:

(A) $(p \land q) \lor (p \lor q)$
(B) $(p \land q) \lor \neg(p \lor q)$
(C) $(p \lor q) \land \neg(p \land q)$
(D) $(p \land q) \land (p \lor q)$

p	**q**	**p XOR q**
F	F	F
F	T	T
T	F	T
T	T	F

Answer at **PollEv.com/cs103** or
text **CS103** to **22333** once to join, then **A**, **B**, or **C**.

Mathematical Implication

Implication

- The → connective is used to represent implications.
  - Its technical name is the ***material conditional*** operator.
- What is its truth table?
- Pull out a sheet of paper, make a guess, and talk things over with your neighbors!

Truth Table for $p \to q$ (implies)

What is the correct truth table for implication? Enter your guess as a list of four values to fill in the rightmost column of the table.
(ex: F, T, ?, F)

p	**q**	**p → q**
F	F	
F	T	
T	F	
T	T	

Answer at **PollEv.com/cs103** or
text **CS103** to **22333** once to join, then your response.

Truth Table for Implication

p	q	$p \rightarrow q$
F	F	T
F	T	T
T	F	F
T	T	T

- Bad bracket, don't get A
- Bad bracket, get A
- Perfect bracket, don't get A
- Perfect bracket, get A

Why This Truth Table?

- The truth values of the $\to$ are the way they are because they're *defined* that way.
- The intuition:
  - Every propositional formula should be either true or false – that's just a guiding design principle behind propositional logic.
  - We want $p \to q$ to be false only when $p \land \neg q$ is true.
  - In other words, $p \to q$ should be true whenever $\neg(p \land \neg q)$ is true.
  - What's the truth table for $\neg(p \land \neg q)$?

Truth Table for Implication

p	q	$p \to q$
F	F	T
F	T	T
T	F	F
T	T	T

The implication is only false if p is true and q isn't. It's true otherwise.

You will need to commit this table to memory. We're going to be using it *a lot* over the rest of the week.

The Biconditional Connective

The Biconditional Connective

- The biconditional connective $\leftrightarrow$ is used to represent a two-directional implication.
- Specifically, $p \leftrightarrow q$ means both that $p \to q$ and that $q \to p$.
- Based on that, what should its truth table look like?
- Take a guess, and talk it over with your neighbor!

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 ⊤ is a value that is always true.
  - The symbol ⊥ 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:

$$(\neg p \to \bot) \to p$$

Operator Precedence

- How do we parse this statement?

$$\neg x \to y \lor z \to x \lor y \land z$$

- Operator precedence for propositional logic:

$$\neg$$
$$\land$$
$$\lor$$
$$\to$$
$$\leftrightarrow$$

- 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

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

The Big Table

Connective	Read As	C++ Version	Fancy Name
¬	"not"	`!`	Negation
∧	"and"	`&&`	Conjunction
∨	"or"	`||`	Disjunction
→	"implies"	*see PS2!*	Implication
↔	"if and only if"	*see PS2!*	Biconditional
⊤	"true"	**`true`**	Truth
⊥	"false"	**`false`**	Falsity

Recap So Far

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

Translating into Propositional Logic

Some Sample Propositions

a: I will be in the path of totality.

b: I will see a total solar eclipse.

"I won't see a total solar eclipse if I'm not in the path of totality."

$$\neg a \to \neg b$$

“p if q”

translates to

$$q \rightarrow p$$

It does *not* translate to

$$p \rightarrow q$$

Some Sample Propositions

a: I will be in the path of totality.

b: I will see a total solar eclipse.

c: There is a total solar eclipse today.

"If I will be in the path of totality, but there's no solar eclipse today, I won't see a total solar eclipse."

$$a \land \neg c \to \neg b$$

“p, but q”

translates to

$$p \land q$$

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 $p \land q$ is false?

Quick Question:

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

De Morgan's Laws

- Using truth tables, we concluded that

$$\neg(p \land q)$$

  is equivalent to

$$\neg p \lor \neg q$$

- We also saw that

$$\neg(p \lor q)$$

  is equivalent to

$$\neg p \land \neg q$$

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

$$\neg(p \land q) \equiv \neg p \lor \neg q$$

- The ≡ symbol is not a connective.
  - The statement $\neg(p \land q) \leftrightarrow (\neg p \lor \neg q)$ 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 $\neg(p \land q) \equiv \neg p \lor \neg q$ means "these two formulas have exactly the same truth table."
- In other words, the notation $\varphi \equiv \psi$ means "φ and ψ 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

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

- Later on, this equivalence will be incredibly useful:

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

Another Important Equivalence

- Here's a useful equivalence. Start with

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

- By De Morgan's laws:

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

$$\equiv \neg p \lor \neg\neg q$$

$$\equiv \neg p \lor q$$

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

Another Important Equivalence

- Here's a useful equivalence. Start with

$$p \rightarrow q \equiv \neg(p \land \neg q)$$

- By De Morgan's laws:

$$p \rightarrow q \equiv \neg(p \land \neg q)$$

$$\equiv \neg p \lor \neg\neg q$$

$$\equiv \neg p \lor$$

- Thus $p \rightarrow q \equiv \neg p \lor q$

> If p is false, then $\neg p \lor q$ 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 $y \geq 8$”

$$x + y = 16 \to x \geq 8 \lor y \geq 8$$

Why All This Matters

- Suppose we want to prove the following statement:

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

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

“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 $x + y \neq 16$. Let x and y be arbitrary numbers such that $x < 8$ and $y < 8$.

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 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.

Next Time

- ***First-Order Logic***
  - Reasoning about groups of objects.
- ***First-Order Translations***
  - Expressing yourself in symbolic math!