Propositional Logic

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

Where We're Going

- ***Propositional Logic*** (Today)
  - Reasoning about Boolean values.
- ***First-Order Logic*** (Wednesday/Friday)
  - Reasoning about properties of multiple objects.

Outline for Today

- ***Propositional Variables***
  - Booleans, math edition!
- ***Propositional Connectives***
  - Linking things together.
- ***Truth Tables***
  - Rigorously defining connectives.
- ***Simplifying Negations***
  - Mechanically computing negations.

Propositional Logic

$$TakeMath51 \lor TakeCME100$$

$$\neg FirstSucceed \to TryAgain$$

$$IsCardinal \land IsWhite$$

$$TakeMath51 \lor TakeCME100$$

$$\neg FirstSucceed \to TryAgain$$

$$IsCardinal \land IsWhite$$

These are ***propositional variables***. Each propositional variable stands for a ***proposition***, something that is either true or false.

$$TakeMath51 \lor TakeCME100$$

$$\neg FirstSucceed \to TryAgain$$

$$IsCardinal \land IsWhite$$

These are ***propositional connectives***, which link propositions into larger propositions

Propositional Variables

- In propositional logic, individual propositions are represented by ***propositional variables***.
- In a move that contravenes programming style conventions, propositional variables are usually represented as lower-case letters, such as *p*, *q*, *r*, *s*, etc.
  - That said, there's nothing stopping you from using multiletter names!
- Each variable can take one one of two values: true or false. You can think of them as **`bool`** values.

Propositional Connectives

- There are seven propositional connectives, five of which will be familiar from programming.
- First, there's the logical "NOT" operation:

$$\neg p$$

- You'd read this out loud as "not *p*."
- The fancy name for this operation is ***logical negation***.

Propositional Connectives

- There are seven propositional connectives, five of which will be familiar from programming.
- Next, there's the logical "AND" operation:

$$\boldsymbol{p \land q}$$

- You'd read this out loud as "*p* and *q*."
- The fancy name for this operation is ***logical conjunction***.

Propositional Connectives

- There are seven propositional connectives, five of which will be familiar from programming.
- Then, there's the logical "OR" operation:

$$p \lor q$$

- You'd read this out loud as "*p* or *q*."
- The fancy name for this operation is ***logical disjunction***. This is an *inclusive* or.

Propositional Connectives

- There are seven propositional connectives, five of which will be familiar from programming.
- There's also the "truth" connective:

$$\top$$

- You'd read this out loud as "true."
- Although this is technically considered a connective, it "connects" zero things and behaves like a variable that's always true.

Propositional Connectives

- There are seven propositional connectives, five of which will be familiar from programming.
- Finally, there's the "false" connective.

$$\bot$$

- You'd read this out loud as "false."
- Like $\top$, this is technically a connective, but acts like a variable that's always false.

Truth Tables

- A ***truth table*** is a table showing the truth value of a propositional logic formula as a function of its inputs.
- Let's go look at the truth tables for the connectives we've seen so far:

$\neg \quad \wedge \quad \vee \quad \top \quad \bot$

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, etc. and the `or` operator in Python.
- If we need an exclusive “or” operator, we can build it out of what we already have.
- Try this yourself! Take a minute to combine these operators together to form an expression that represents the exclusive or of p and q (something that’s true if and only if exactly one of p and q are true.)

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 $\boldsymbol{p \lor q}$ is false?

de Morgan's Laws

$\neg(p \wedge q)$ ***is equivalent to*** $\neg p \vee \neg q$

$\neg(p \vee q)$ ***is equivalent to*** $\neg p \wedge \neg q$

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: if `p()` returns false, `q()` is never evaluated.)

Mathematical Implication

Implication

- We can represent implications using this connective:

$$\boldsymbol{p \to q}$$

- You'd read this out loud as "p implies q."
  - The fancy name for this is the ***material conditional***.
- ***Question:*** What should the truth table for $p \to q$ look like?
- Pull out a sheet of paper, make a guess, and talk things over with your neighbors!

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

An implication is false only when the antecedent is true and the consequent is false.

Every formula is either true or false, so these other entries have to be true.

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

Important observation:

The statement $p \to q$ is true whenever $p \wedge \neg q$ is false.

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

An implication with a false antecedent is called ***vacuously true***.

An implication with a true consequent is called ***trivially true***.

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

Please commit this table to memory. We're going to need it, extensively, over the next couple of weeks.

Fun Fact 1: The Contrapositive Revisited

Fun Fact 2: Proof by Contradiction

Fun Fact 3: Implies, Another Way

An Important Equivalence

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

$$\boldsymbol{p \to q} \quad \textbf{\textit{is equivalent to}} \quad \boldsymbol{\neg(p \land \neg q)}$$

- Later on, this equivalence will be incredibly useful:

$$\boldsymbol{\neg(p \to q)} \quad \textbf{\textit{is equivalent to}} \quad \boldsymbol{p \land \neg q}$$

Another Important Equivalence

- Here's a useful equivalence. Start with

$$p \to q \quad \textit{is equivalent to} \quad \neg(p \land \neg q)$$

- By de Morgan's laws:

$$p \to q \quad \textit{is equivalent to} \quad \neg(p \land \neg q)$$

$$\textit{is equivalent to} \quad \neg p \lor \neg\neg q$$

$$\textit{is equivalent to} \quad \neg p \lor q$$

- Thus $p \to q$ is equivalent to $\neg p \lor q$

The Biconditional Connective

The Biconditional Connective

- On Friday, we saw that "*p* if and only if *q*" means both that $p \to q$ and $q \to p$.
- We can write this in propositional logic using the ***biconditional*** connective:

$$p \leftrightarrow q$$

- This connective's truth table has the same meaning as "*p* implies *q* and *q* implies *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 $\leftrightarrow$ is to think of it as equality: the two propositions must have equal truth values.

Negating a Biconditional

- How do we simplify

$$\neg(p \leftrightarrow q)$$

  using the tools we've seen so far?

- There are many options, but here are our two favorites:

$$p \leftrightarrow \neg q \qquad\qquad \neg p \leftrightarrow q$$

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

- The main points to remember:
  - $\neg$ binds to whatever immediately follows it.
  - $\wedge$ and $\vee$ bind more tightly than $\rightarrow$.
- 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? ***Please ask!***

The Big Table

Connective	Read Aloud As	C++ Version	Fancy Name
$\neg$	“not”	`!`	Negation
$\wedge$	“and”	`&&`	Conjunction
$\vee$	“or”	`||`	Disjunction
$\top$	“true”	`true`	Truth
$\bot$	“false”	`false`	Falsity
$\rightarrow$	“implies”	*see PS2!*	Implication
$\leftrightarrow$	“if and only if”	*see PS2!*	Biconditional

Time-Out for Announcements!

Ask Me Anything

- Between lectures, I'll make a post on EdStem soliciting your questions.
- This is a chance to ask questions about any topic: course material, life advice, music recommendations, etc.
  - Just keep things civil and only ask questions if you really want to hear the answer.
- Feel free to questions you like. I'll take some of the more popular questions in lecture.

Office Hours

- Office hours start today. Think of them as “drop-in help hours” where you can ask questions on problem sets, lecture topics, etc.
  - Check the Guide to Office Hours on the course website for the schedule.
- Most office hours are held in person in the Huang basement. A few are purely online. Mine are in my office, Durand 317.
- Once you arrive, sign up on QueueStatus so that we can help people in the order they arrived:

  https://queuestatus.com/queues/782
- Office hours are much less crowded earlier in the week than later.

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: $p \land q$
  - Disjunction: $p \lor q$
  - Truth: $\top$
  - Falsity: $\bot$
  - Implication: $p \to q$
  - Biconditional: $p \leftrightarrow q$

Negation Practice

- Here's a propositional formula that contains some negations. Simplify it as much as possible:

$$\neg(p \land q \to r \lor s)$$

Negation Practice

- Here's a propositional formula that contains some negations. Simplify it as much as possible:

$$p \land q \land \neg r \land \neg s$$

Negation Practice

- Here's a propositional formula that contains some negations. Simplify it as much as possible:

$$\neg\big((p \lor (q \land r)) \leftrightarrow (a \land b \land c \to d)\big)$$

Negation Practice

- Here's a propositional formula that contains some negations. Simplify it as much as possible:

$$(p \lor (q \land r)) \leftrightarrow (a \land b \land c \land \neg d)$$

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 $y \geq 8$”

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

Why All This Matters

- Suppose we want to prove the following statement:

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

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

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: We will prove the contrapositive, namely, that if $x < 8$ and $y < 8$, then $x + y \neq 16$.

Pick x and y where $x < 8$ and $y < 8$. We want to show that $x + y \neq 16$. 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 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!