Propositional Logic

Logistics: Lecture Participation

Lecture Participation

- Starting Wednesday, we will be using the website PollEV to ask questions in lecture for attendance credit.
- If you answer these questions in lecture, you'll get attendance credit for the day.
 - You don't need to have the right answers you just need to respond to the questions.
- If you don't attend lecture in person, no worries! You can alternatively answer questions about the lecture on Gradescope before the start of the next lecture to earn participation credit.
 - You do have to get the answers right, though you get unlimited resubmit attempts.
- If you don't do either, that's fine too! We'll count your final exam score in place of your participation grade.

Lecture Participation

- We'll dry-run PollEV questions today.
- Let's start with the following warm-up:

Make a music recommendation!

Answer at https://cs103.stanford.edu/pollev

- Here's a few music recs of my own:
 - *Digitonium* by Turkuaz.
 - Timeless by Jami Sieber.
 - Little Big and Little Big II by Aaron Parks.
 - Nurture by Porter Robinson.

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 v TakeCME100

 $\neg FirstSucceed \rightarrow TryAgain$

IsCardinal \(\Lambda \) IsWhite

TakeMath51 v TakeCME100

 $\neg FirstSucceed \rightarrow TryAgain$

IsCardinal \(\) IsWhite

These are *propositional*variables. Each propositional

variable stands for a

proposition, something that is

either true or false.

TakeMath51 v TakeCME100

 $\neg FirstSucceed \rightarrow TryAgain$

IsCardinal \(\ \ 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.

- 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.

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

$p \wedge q$

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

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

p v 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.

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

Т

- 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.

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

丄

- You'd read this out loud as "false."
- Like T, 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 \wedge \vee \top \bot

Inclusive and Exclusive OR

- The v connective is an *inclusive* "or." It's true if at least one of the operands is true.
 - It's similar to the || operator in C, C++, Java, etc. and the or operator in Python.
- Sometimes we need an *exclusive* "or," which isn't true if both inputs are true.
- We can build this out of what we already have.

Write a propositional logic formula for the exclusive OR of p and q.

Answer at https://cs103.stanford.edu/pollev

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 $p \ v \ q$ is false?

de Morgan's Laws

$$\neg(p \land q)$$
 is equivalent to $\neg p \lor \neg q$

$$\neg(p \lor q)$$
 is equivalent to $\neg p \land \neg q$

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

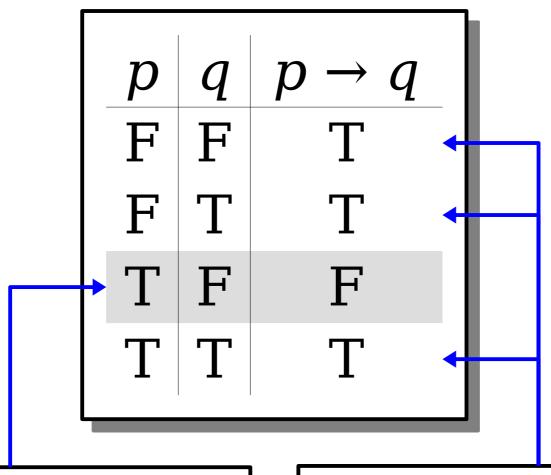
Implication

We can represent implications using this connective:

$$p \rightarrow q$$

- 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 $p \rightarrow q$ look like?

p	q	$p \rightarrow q$	
F	F	Τ	
F	T	T	
T	F	F	
T	T	T	
· · ·			



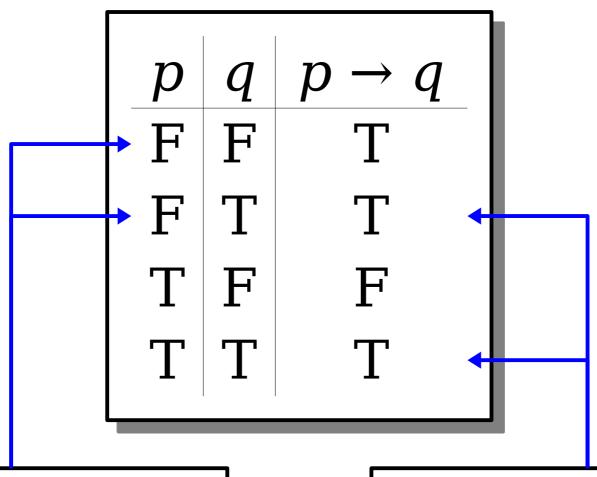
An implication is false only when the antecedent is true and the consequent is false.

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

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

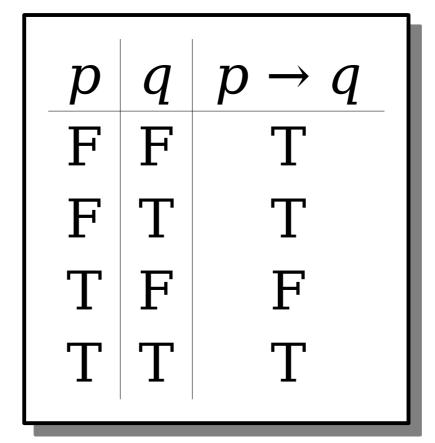
Important observation:

The statement $p \rightarrow q$ is true whenever $p \land \neg q$ is false.



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

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



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

An Important Equivalence

• The truth table for for $p \rightarrow q$ is chosen so that the following is true:

$$p \rightarrow q$$
 is equivalent to $\neg (p \land \neg q)$

• Later on, this equivalence will be incredibly useful:

```
\neg (p \rightarrow q) is equivalent to p \land \neg q
```

The Biconditional Connective

The Biconditional Connective

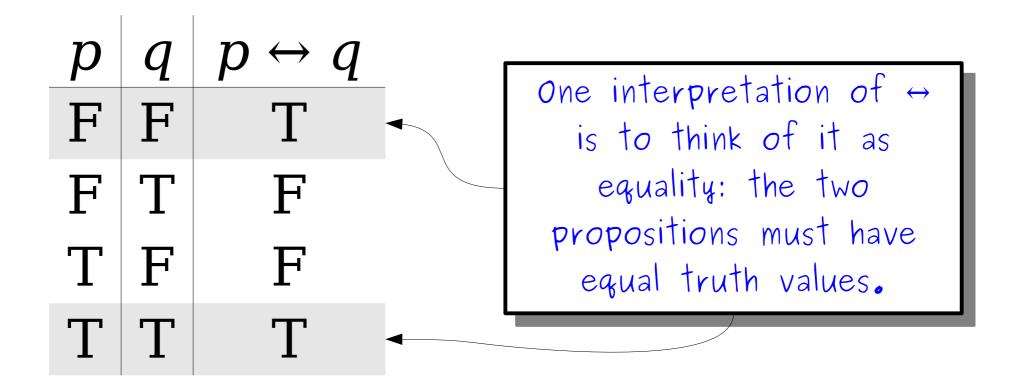
- In our previous lecture, we saw that the statement "p if and only if q" means both that $p \rightarrow q$ and $q \rightarrow p$.
- We can write this in propositional logic using the *biconditional* connective:

$p \leftrightarrow q$

- 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?

Biconditionals

- The biconditional connective $p \leftrightarrow q$ has the same truth table as $(p \rightarrow q) \land (q \rightarrow p)$.
- Here's what that looks like:



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:

$$p \leftrightarrow \neg q$$

$$\neg p \leftrightarrow q$$

Question to ponder: what is the truth table for these statements, and where have you seen it before?

Operator Precedence

How do we parse this statement?

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

Operator precedence for propositional logic:

∧ ∨ → ↔

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

Operator Precedence

How do we parse this statement?

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

Operator precedence for propositional logic:

∧ ∨ → ↔

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

Operator Precedence

- The main points to remember:
 - ¬ binds to whatever immediately follows it.
 - Λ and V bind more tightly than \rightarrow .
- We will commonly write expressions like $p \land q \rightarrow r$ 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
7	"not"		Negation
٨	"and"	&&	Conjunction
V	"or"		Disjunction
Т	"true"	true	Truth
上	"false"	false	Falsity
\rightarrow	"implies"	see PS2!	Implication
\leftrightarrow	"if and only if"	see PS2!	Biconditional

Time-Out for Announcements!

Submitting Work

- All assignments should be submitted through GradeScope.
 - The programming portion of the assignment is submitted separately from the written component.
 - The written component *must* be typed; handwritten solutions don't scan well and get mangled in GradeScope.
- All assignments are due at 3:00PM. You have three "late days" you can use throughout the quarter. Each automagically extends assignment deadlines from Friday at 3:00PM to Saturday at 3:00PM; at most one late day can be used per assignment.
 - *Very good idea:* Leave at least two hours buffer time for your first assignment submission, just in case something goes wrong.
 - Very bad idea: Wait until the last minute to submit.
- Your score on the problem sets is the square root of your raw score. So an 81% maps to a 90%, a 50% maps to a 71%, etc. This gives a huge boost even if you need to turn something in that isn't done.

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.
- TA office hours are held in person in the Huang basement. Alex's are in Gates 490. Keith's are in 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. Stop by on Monday and Tuesday!

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 v q
 - Truth: T
 - Falsity: ⊥
 - Implication: $p \rightarrow q$
 - Biconditional: $p \leftrightarrow q$

Why All This Matters

Why All This Matters

 Suppose we want to prove the following statement:

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

$$x + y = 16 \rightarrow x \ge 8 \ \forall \ y \ge 8$$

Why All This Matters

 Suppose we want to prove the following statement:

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

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

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

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

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

Pick x and y where x < 8 and y < 8. We want to show that $x + y \ne 16$. To see this, note that

$$x + y < 8 + y$$

 $< 8 + 8$
 $= 16.$

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

Why This Matters

- Propositional logic lets us symbolically manipulate statements and theorems.
 - This can help us better understand what a theorem says or what a definition means.
- It's also very useful for proofs by contradiction and contrapositive.
- Being able to negate statements mechanically can reduce the likelihood of taking an negation of contrapositive wrong.

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

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

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

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

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

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

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

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

Next Time

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