

#### Introduction

- Propositional Logic
  - · Reasoning about Boolean values.
- First-Order Logic
  - Reasoning about properties of multiple objects.

#### Outline

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

# TakeMath51 V TakeCME100 ¬FirstSucceed → TryAgain IsCardinal ∧ IsWhite

#### TakeMath51 V TakeCME100

¬FirstSucceed → TryAgain

IsCardinal \( \) IsWhite

These are **propositional**variables. Each propositional

variable stands for a

proposition, something that is

either true or false.

#### TakeMath51 V TakeCME100

¬FirstSucceed → TryAgain

IsCardinal \( \Lambda \) 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:

### $p \vee q$

- 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:

# Summary of Important Points

- The V 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.

#### 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 V q is false?



#### de Morgan's Laws



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

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

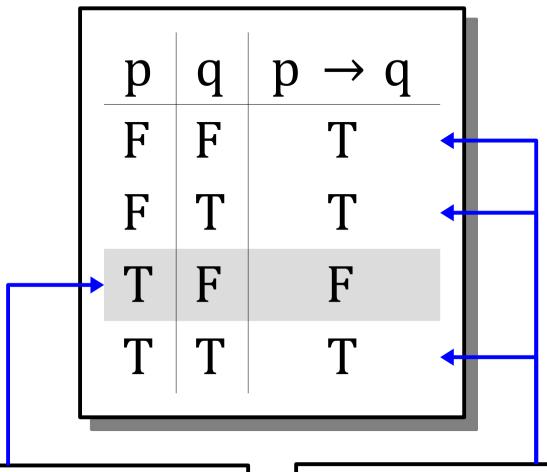
# Mathematical Implication

# 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 → q look like?
- Pull out a sheet of paper, make a guess, and talk things over with your neighbors!



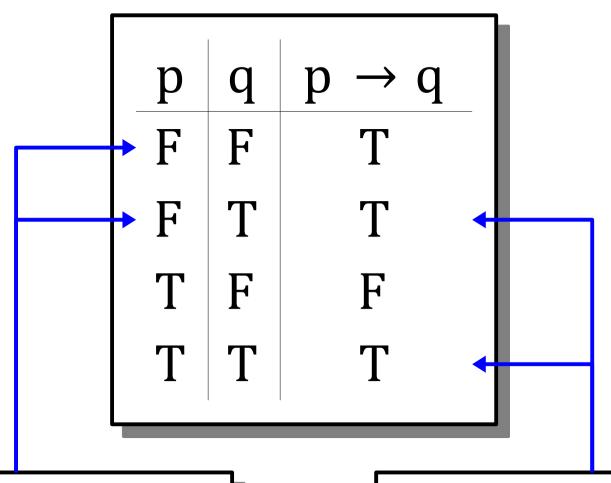
An implication is false only when the antecedent is true and the consequent is false.

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*.

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

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

# An Important Equivalence

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

$$m{p} o m{q}$$
 is equivalent to  $\neg (m{p} \land \neg m{q})$ 

 Later on, this equivalence will be incredibly useful:

$$eg(oldsymbol{p} o oldsymbol{q})$$
 is equivalent to  $oldsymbol{p} \wedge 
eg oldsymbol{q}$ 

# Another Important Equivalence

· Here's a useful equivalence. Start with

$$m{p} 
ightarrow m{q}$$
 is equivalent to  $\neg (m{p} \land \neg m{q})$ 

• By de Morgan's laws:

• Thus  $p \rightarrow q$  is equivalent to  $\neg p \lor q$ 

#### The Biconditional Connective

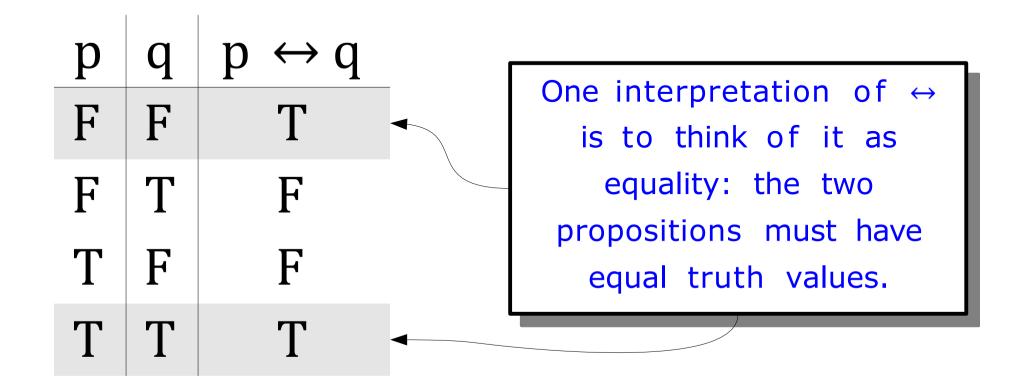
- On Friday, we saw that "p if and only if q" means both that  $p \rightarrow q$  and  $q \rightarrow p$ .
- We can write this in propositional logic using the biconditional connective:

#### $p \leftrightarrow q$

- 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?
- Take a guess, and talk it over with your neighbor!

#### Biconditionals

- Here's its truth table:



# Negating a Biconditional

How do we simplify

$$\neg (\boldsymbol{p} \leftrightarrow \boldsymbol{q})$$

using the tools we've seen so far?

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

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

# 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:

Λ

V

 $\rightarrow$ 

 $\leftrightarrow$ 

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

# Operator Precedence

- The main points to remember:
  - ¬ binds to whatever immediately follows it.
  - $\land$  and  $\lor$  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.

•

# The Big Table

Connective	Read Aloud As	C++ Version	Fancy Name
٦	"not"		Negation
Λ	"and"	&&	Conjunction
V	"or"		Disjunction
Т	"true"	true	Truth
Т	"false"	false	Falsity
$\rightarrow$	"implies"	see PS2!	Implication
$\leftrightarrow$	"if and only if"	see PS2!	Biconditional

## Recap So Far

- A propositional variable is a variable that is either true or false.
- · The propositional connectives are
  - Negation: ¬p
  - Conjunction: p ∧ q
  - Disjunction: p V q
  - Truth: T
  - Falsity: ⊥
  - Implication:  $p \rightarrow q$
  - Biconditional:  $p \leftrightarrow q$

# 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.