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.

Propositional Logic

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

Some Sample Propositions

- I am not throwing away my shot.
- I’m just like my country.
- I’m young, scrappy, and hungry.
- I’m not throwing away my shot.
- I’m ‘a get a scholarship to King’s College.
- I prob’ly shouldn’t brag, but dag, I amaze and astonish.
- The problem is I got a lot of brains but no polish.

Things That Aren't Propositions

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

- There are seven propositional connectives, many 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, many 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***.

Propositional Connectives

- There are seven propositional connectives, many 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.

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 three connectives we've seen so far:

¬ ∧ ∨

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.

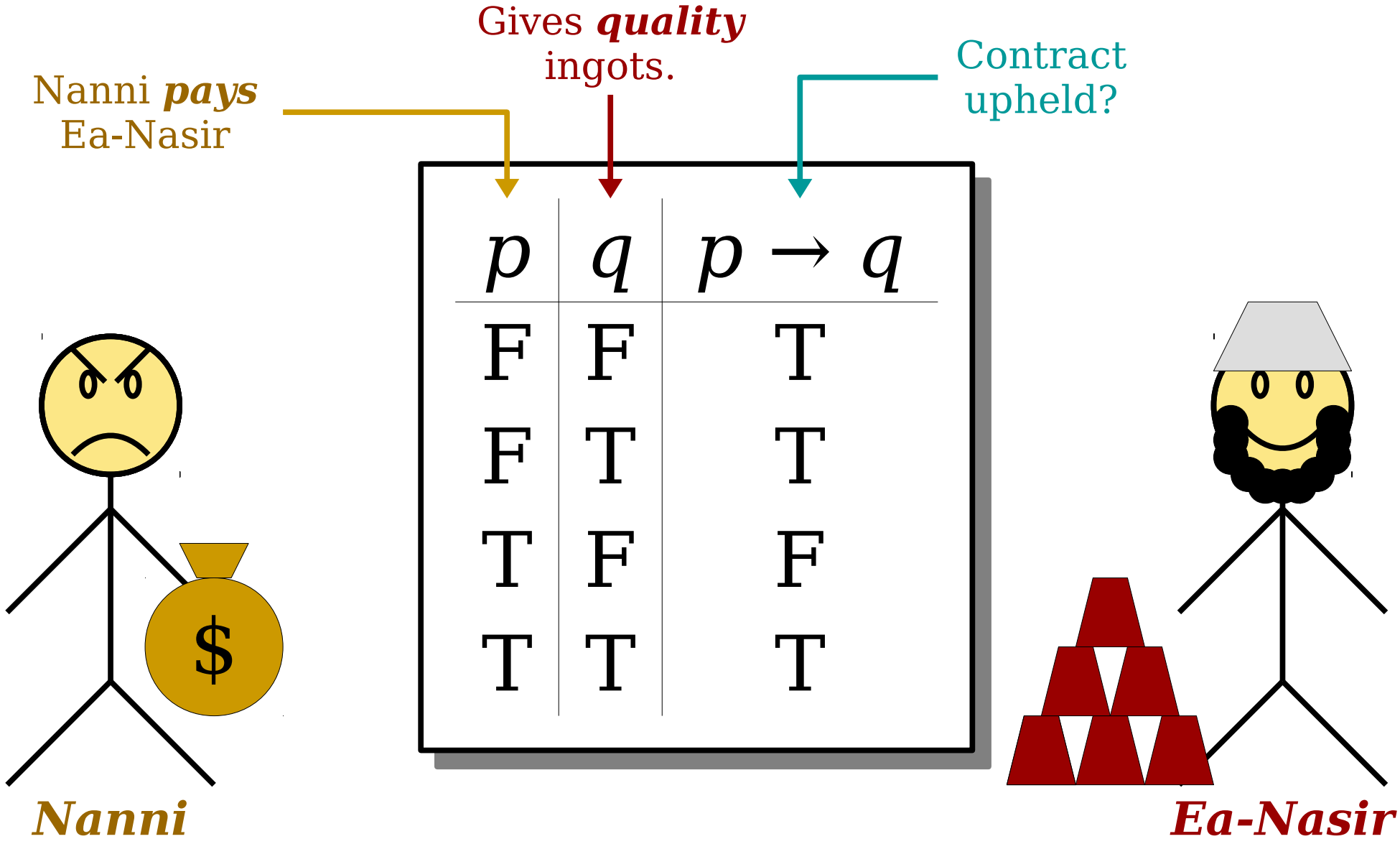
Mathematical Implication

Implication

- We can represent implications using this connective:

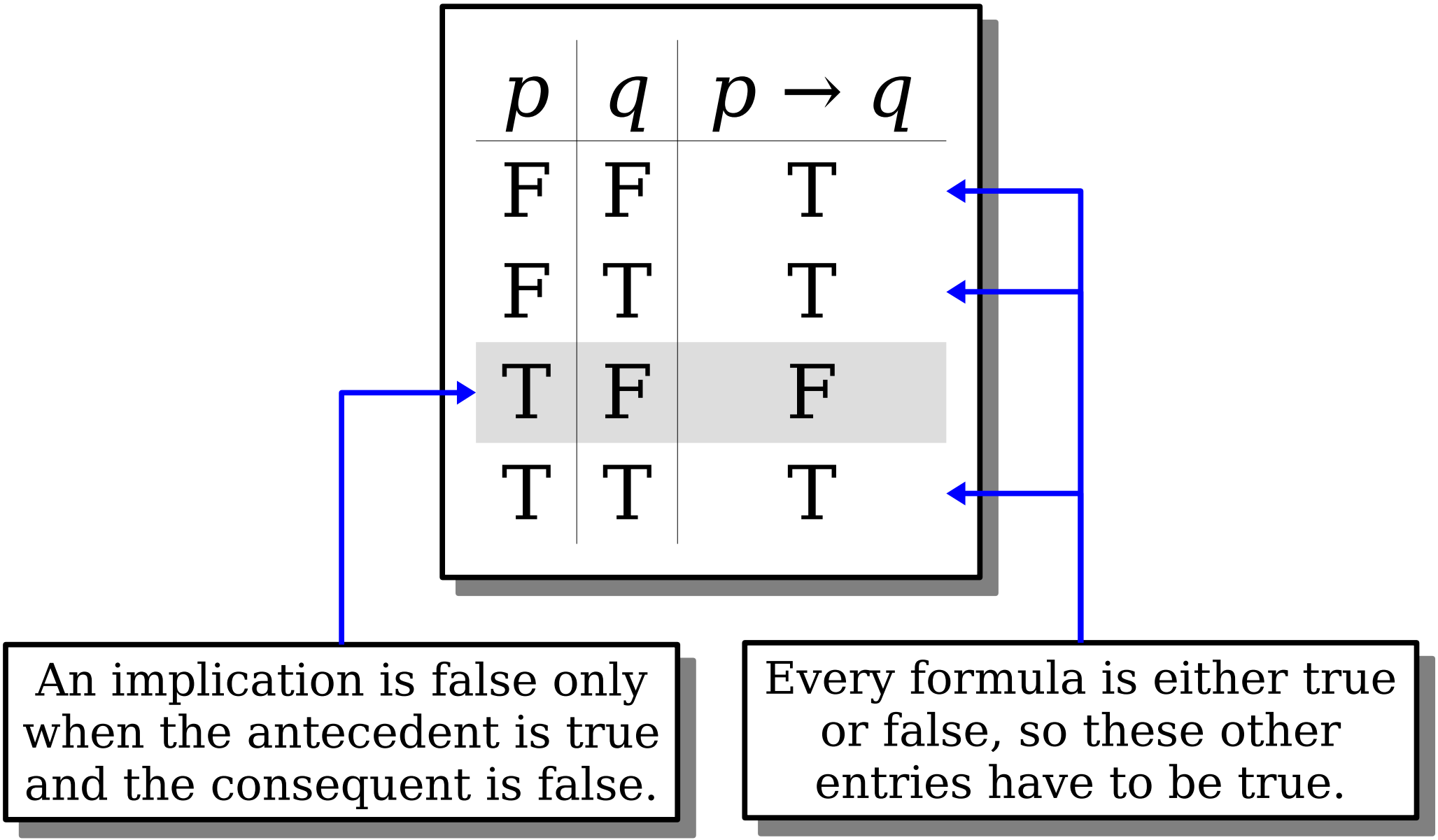
$$\boldsymbol{p \rightarrow q}$$

- 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?
- Pull out a sheet of paper, make a guess, and talk things over with your neighbors!



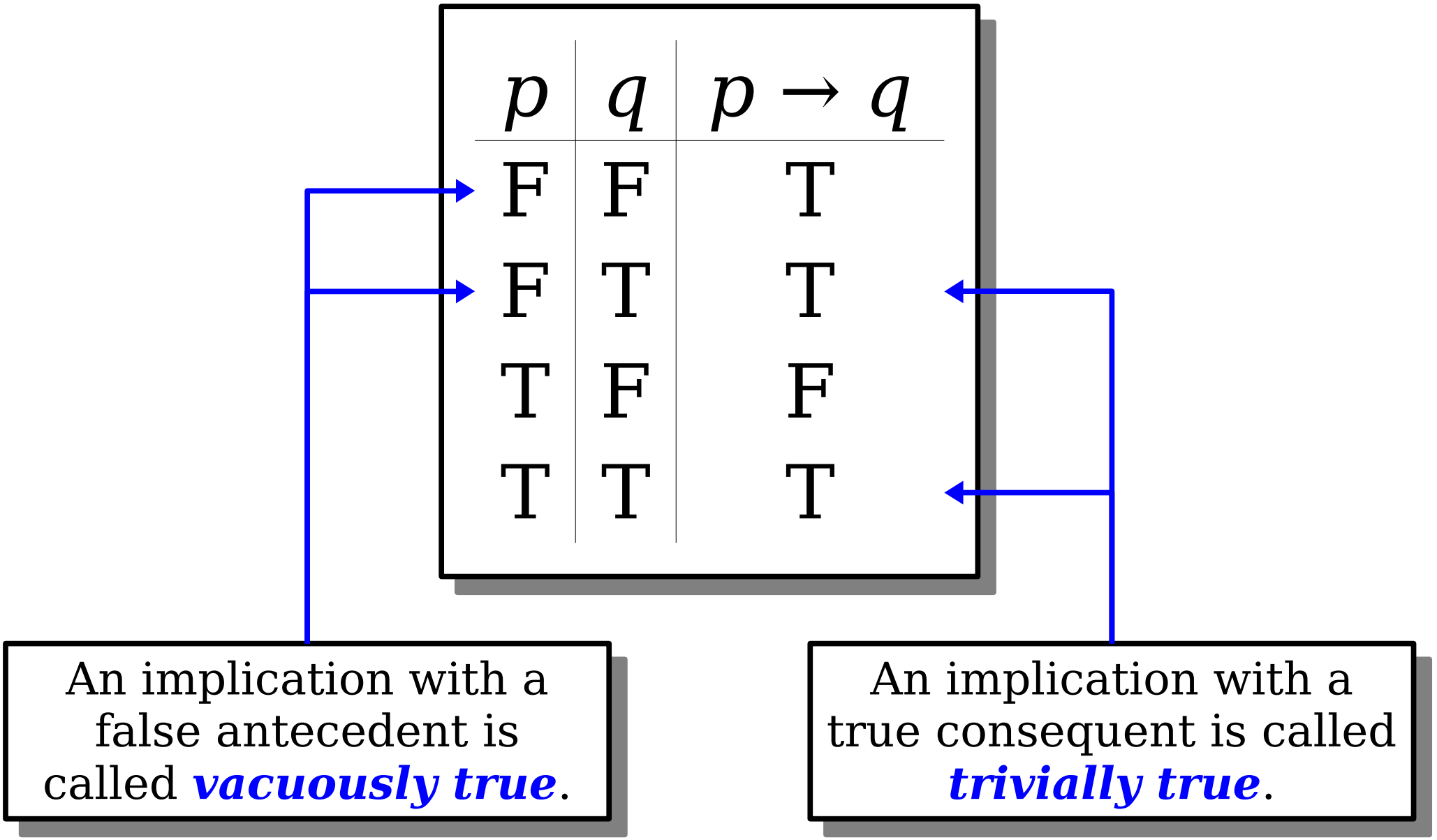
Ancient Babylonian Contract:

If Nanni pays money to Ea-Nasir, then
Ea-Nasir will give Nanni quality copper ingots.


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 $p \rightarrow q$
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: The Contrapositive Revisited

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:

$$\boldsymbol{p \leftrightarrow q}$$

- 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 ↔ 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:

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

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?

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

- 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? ***Please ask!***

The Big Table

Connective	Read Aloud 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

Time-Out for Announcements!

Problem Set One

- The checkpoint problem for PS1 was due at 2:30PM today.
  - We'll try to have it graded and returned by Wednesday morning.
  - Solutions are available. ***Please read them!***
- The remaining problems from PS1 are due on Friday at 2:30PM.
  - Have questions? Stop by office hours or ask on Piazza!

Your Questions

“What do you see as the most important area of CS right now?”

From my vantage point, I'd say the biggest question is "how do we ensure that everyone who wants to use technology to solve problems in their world has the ability to do so?" The potential impact of using computing power is huge, and no one person knows all the issues people face worldwide. Computers can't solve every problem, but they can help immensely in many arenas, and the question is how to get these skills out to as many people as possible from as broad a spectrum as possible.

“If there are most certainly more problems to solve than programs to solve them, what does this mean for the limits of computability and of human knowledge?”

This question goes really deep, and I'd like to return to it later in the quarter when we talk about undecidability and unrecognizability. Ask me this again once we've talked about self-reference!

“What are your tips for when you first approach a proof problem? I often find myself confused as to how to get started”

It's totally normal to feel that way! If there were a single, unified approach that always worked, I promise I'd tell you what it was. Alas, there isn't one so we'll have to resort to heuristics.

The main thing to keep in mind is that you aren't expected to see things instantly and on the first time around. Plan to throw away beautiful arguments and lines of reasoning, and be open to the idea that you'll have to back up and start over.

It never hurts to write out, explicitly, what you're assuming and what you need to prove. It never hurts to try out concrete examples and to draw pictures. It's good to look at other proofs to see if you can draw inspiration from them. Don't be afraid to pause one line of exploration to explore another. You'll build up an intuition for what works as you get more experience. So hang in there, and good luck!

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 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."

$$\neg a \rightarrow \neg b$$

“***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."

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

“***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 "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

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

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

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 $y \geq 8$"

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

Why All This Matters

- Suppose we want to prove the following statement:

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

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

"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: We will prove the contrapositive of this statement: 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!