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** (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

- 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.
Más Proposiciones

- Tú, tú eres el imán y yo soy el metal.
- Me voy acercando y voy armando el plan.
- Sólo con pensarlo se acelera el pulso.
- Ya, ya me está gustando más de lo normal.
- Todos mis sentidos van pidiendo más.
- Esto hay que tomarlo sin ningún apuro.
Things That Aren't Propositions

Commands cannot be true or false.
Things That Aren't Propositions

Questions cannot be true or false.
Things That Aren't Propositions

The first half is a valid proposition.

I am the walrus, goo goo g'joob

Jibberish 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 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:** \( p \land q \)
  - Read “\( p \) and \( q \)”
  - \( p \land q \) is true if and only if both \( p \) and \( q \) are true.
  - Also called *logical conjunction*.

- **Logical OR:** \( p \lor q \)
  - Read “\( p \) or \( q \)”
  - \( p \lor q \) 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 $\lor$ 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.
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!
Why This Truth Table?

- The truth values of the → 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 $p \rightarrow q$ to be false only when $p \land \neg q$ is true.
  - In other words, $p \rightarrow q$ 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

<table>
<thead>
<tr>
<th>$p$</th>
<th>$q$</th>
<th>$p \rightarrow q$</th>
</tr>
</thead>
<tbody>
<tr>
<td>F</td>
<td>F</td>
<td>T</td>
</tr>
<tr>
<td>F</td>
<td>T</td>
<td>T</td>
</tr>
<tr>
<td>T</td>
<td>F</td>
<td>F</td>
</tr>
<tr>
<td>T</td>
<td>T</td>
<td>T</td>
</tr>
</tbody>
</table>

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

• The **biconditional** connective $p \iff q$ is read “$p$ if and only if $q$.”

• Here's its truth table:

\[
\begin{array}{ccc}
  p & q & p \iff q \\
  \hline
  F & F & T \\
  F & T & F \\
  T & F & F \\
  T & T & T \\
\end{array}
\]

One interpretation of $\iff$ is to think of it as **equality**: the two propositions must have equal truth values.
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:
  \[
  (\neg p \rightarrow \bot) \rightarrow p
  \]
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:
  \[ \neg \rightarrow \land \lor \]

- 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:

\[
\begin{align*}
\neg & \\
\land & \\
\lor & \\
\to & \\
\leftrightarrow & 
\end{align*}
\]

• 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.
  • $\land$ 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.
• Confused? Just ask!
# The Big Table

<table>
<thead>
<tr>
<th>Connective</th>
<th>Read As</th>
<th>C++ Version</th>
<th>Fancy Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>¬</td>
<td>“not”</td>
<td>!</td>
<td>Negation</td>
</tr>
<tr>
<td>∧</td>
<td>“and”</td>
<td>&amp;&amp;</td>
<td>Conjunction</td>
</tr>
<tr>
<td>∨</td>
<td>“or”</td>
<td></td>
<td></td>
</tr>
<tr>
<td>→</td>
<td>“implies”</td>
<td><em>see PS2!</em></td>
<td>Implication</td>
</tr>
<tr>
<td>↔</td>
<td>“if and only if”</td>
<td><em>see PS2!</em></td>
<td>Biconditional</td>
</tr>
<tr>
<td>⊤</td>
<td>“true”</td>
<td>true</td>
<td>Truth</td>
</tr>
<tr>
<td>⊥</td>
<td>“false”</td>
<td>false</td>
<td>Falsity</td>
</tr>
</tbody>
</table>
Time-Out for Announcements!
Join us in rebranding what it means to be a technologist.

http://tinyurl.com/sheplusplusteam

Deadline is Sunday, October 8 at 11:59 PM

Message @ShePlusPlus on Facebook with any questions
Join the team that organizes Stanford's national hackathon!

TreeHacks brings together hundreds of collegiate hackers from around the world for 36 hours of non-stop building. Join the team that makes this event happen.

APPLY HERE
Applications close October 4th at 11:59 pm

LEARN MORE @ INFO SESSION HERE (new time and location!)
Sunday October 1st, 9-9:30 pm in Old Union 215
Want to be more involved in LGBTQ+ leadership on campus?
Care deeply about queer issues in STEM fields?

Come on over to the one-and-only
~*~ oSTEM Undergrad Mixer! ~*~

WHO: Stanford oSTEM is a chapter of the national organization, Out in Science, Technology, Engineering and Mathematics, dedicated to serving gender minority students in the STEM fields, with the primary goal of fostering success in leadership, academic pursuits, and professional activity.

WHEN/WHERE: WEDNESDAY, 10/4 at 7 pm at the **QSpot** (formerly known as the LGBT-CRC)!!!

WHAT: Join your fellow LGBTQ+ undergrads in STEM for the undergraduate oSTEM mixer!
We'll be playing uncomfortable icebreakers, meeting and mingling, and discussing oSTEM activities for this quarter (bonding, mentorship, study nights!). Come on over for excellent company and ~FREE PIZZA!~

RSVP **HERE**! (so we know how much food to order)
The applications for the Frosh Internship Program are now open!!

The Frosh Internship Program is a great opportunity for you to be more involved with SOLE where you will get to shadow an officer position of your preference in the fall and another in the winter. You will attend officer meetings, help out with society events, and carry out any projects ideas you want to bring to SOLE, all while having the officer core to support you from beginning to end. You will get to work with the other interns on fun projects such as planning out one of the weekly meetings with food/activity of your choice. The internship provides great leadership experience, and many interns actually go on to become SOLE officers. We're looking for people who are passionate and willing to commit time and effort into helping SOLE grow into a bigger and better familia.

Download the application here and email it to dm628@stanford.edu and camilah@stanford.edu by Saturday, October 7th at 7PM. They’ve requested that you name the file FirstNameLastNameSOLEFroshApp.doc for simplicity.
Interested in computer science, education, or diversity in STEM? Join Girls Teaching Girls to Code (GTGTC) leadership! To start off this school year, we are currently accepting applications to join us on the executive team. You’ll get to be an inspiration to hundreds of girls, lead GTGTC workshops and events, and meet other women in CS to build your network.

No prior CS experience is necessary - freshmen are encouraged to apply!

Applications are due by Friday, October 6, 2017 at 5:00 PM.

For more information, check out our website at http://girlsteachinggirlstocode.org/. You can also reach out to us via email with any questions.

We hope that you can join us in leading this empowering and enriching initiative. Good luck with this new school year, and welcome aboard!
Problem Set One

• The checkpoint problem for PS1 was due at 2:30PM today.
  • We'll try to have it graded and returned by tomorrow evening.

• The remaining problems from PS1 are due on Friday at 2:30PM.
  • Have questions? Stop by office hours, or ask on Piazza, or email the staff list!
Your Questions
“In your opinion, what's the coolest concept in programming (that we'd be able to understand)?”

Everything is made of pointers, arrays, and numbers.

It's crazy how many ways you can combine these concepts together, and it's one of the reasons I love studying data structures so much. You'd think we'd have figured out every way of combining these ideas, and yet we keep still discovering new ones.
“How should I prepare for technical interviews?”

I’d definitely recommend grabbing a copy of “Cracking the Coding Interview” by Gayle McDowell. Her advice about both the technical portions and the nontechnical portions of the interview are extremely valuable.

If you’re an on-campus student, take CS9! Cynthia and Jerry have excellent advice about how to prepare and can help you practice.

And get lots of practice! Doing a little something every day compounds very quickly.

Don’t just memorize solutions. That shows. Instead, understand them and see where they come from.

And be prepared to talk about your resume! Technical interviews aren’t the SAT.
“Can I write my solutions and scan it through CamScanner and then submit the PDF through GradeScope? Or is it required that the solutions be written?”

We do ask that you type up solutions. This is new this quarter and stems from a lot of trouble we’ve had in the past with GradeScope mangling handwritten submissions. We’re putting together a LaTeX workshop to teach you how to do this easily and beautifully; stay tuned for details!
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: \( p \lor q \)
  
  • 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.

\[ \neg a \rightarrow \neg b \]

“I won’t see a total solar eclipse if I’m not in the path of totality.”
“p if q”
translates to

q → p

It does not translate to

p → 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.

“\(a \land \neg c \rightarrow \neg b\)\n
"If I will be in the path of totality, but there's no solar eclipse today, I won't see a total solar eclipse."
“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 \(\equiv\) 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 “\(\varphi\) and \(\psi\) always have the same truth values, regardless of how the variables are assigned.”
An Important Equivalence

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

\[
p \rightarrow q \quad \equiv \quad \neg (p \land \neg q)
\]

• Later on, this equivalence will be incredibly useful:

\[
\neg (p \rightarrow q) \quad \equiv \quad p \land \neg q
\]
Another Important Equivalence

• Here's a useful equivalence. Start with
  \[ p \rightarrow q \equiv \neg(p \land \neg q) \]

• By De Morgan's laws:
  \[ p \rightarrow q \equiv \neg(p \land \neg q) \]
  \[ \equiv \neg p \lor \neg q \]
  \[ \equiv \neg p \lor q \]

• Thus \[ p \rightarrow q \equiv \neg p \lor q \]
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:

$$p \rightarrow q \equiv \neg q \rightarrow \neg p$$
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 \( y \geq 8 \)"

\[
x + y = 16 \rightarrow x \geq 8 \lor y \geq 8
\]
Why All This Matters

• Suppose we want to prove the following statement:

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

  \[ \neg(x \geq 8 \lor y \geq 8) \rightarrow \neg(x + y = 16) \]
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:** By contrapositive. We will prove that if $x < 8$ and $y < 8$, then $x + y \neq 16$. To see this, note that

\[
x + y < 8 + y \\
< 8 + 8 \\
= 16
\]

This means that $x + y < 16$, so $x + y \neq 16$, which is what we needed to show. ■
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 $y \geq 8$”

  $\neg (x + y = 16 \rightarrow x \geq 8 \lor y \geq 8)$
Why All This Matters

• Suppose we want to prove the following statement:

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

  
  \[ x + y = 16 \land \neg(x \geq 8 \lor y \geq 8) \]
Why All This Matters

• Suppose we want to prove the following statement:

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

\[
\begin{align*}
  x + y &= 16 \land x < 8 \land y < 8 \\
  \text{“} x + y = 16, \text{ but } x < 8 \text{ and } y < 8. \text{”}
\end{align*}
\]
**Theorem:** If $x + y = 16$, then $x \geq 8$ or $y \geq 8$.

**Proof:** Assume for the sake of contradiction that $x + y = 16$, but that $x < 8$ and $y < 8$. Then

\[
x + y < 8 + y \\
< 8 + 8 \\
= 16
\]

So $x + y < 16$, contradicting that $x + y = 16$. We have reached a contradiction, so our assumption must have been wrong. Therefore if $x + y = 16$, then $x \geq 8$ or $y \geq 8$. ■
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!