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'd prefer not to attend lectures, that's okay! You can opt to count your final exam in place of participation.
  - We'll send out a form where you can opt-out of participation in Week 4.
- CGOE students: We automatically opt you out of participation, since we assume you aren't physically here.

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***

Click "Register" and enter your Stanford e-mail to get to the SUNet login page.

- Here are a few music recs of our own:
  - Jami Sieber - *Timeless*.
  - Aaron Parks - *Little Big* and *Little Big II*.
  - Arthur Moon - NPR Music Tiny Desk Concert.
  - Shakey Graves – *Roll the Bones* (check out *Audiotree Live* version).

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***.
- 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***.

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 examine the truth tables for the connectives we're exploring today!

"I don't love cupcakes."

LoveCupcakes : I love cupcakes.

¬***LoveCupcakes***

Propositional Variables

- In propositional logic, individual propositions are represented by ***propositional variables***.
- Each variable can take one one of two values: true or false. You can think of them as `bool` values.
- 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!

"I don't love cupcakes."

LoveCupcakes : I love cupcakes.

¬***LoveCupcakes***

Propositional Connectives

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

$$p \land q$$

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

"It's cardinal and white."

IsCardinal : It's cardinal.

IsWhite : It's white.

IsCardinal ∧ ***IsWhite***

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.

"You must take Math 51 or CME 100."

TakeMath51 : You must take Math 51.

TakeCME100 : You must take CME 100.

TakeMath51 ∨ ***TakeCME100***

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.

Inclusive and Exclusive OR

- The ∨ 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 $\boldsymbol{p \land q}$ 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

$\neg(p \land q)$ ***is equivalent to*** ??

$\neg(p \lor q)$ ***is equivalent to*** ??

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.)

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?

Mathematical Implication

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

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 $p \land \neg q$ 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.

"If at first you don't succeed, try again."

FirstSucceed : You succeed at first.

TryAgain : You ought to try again.

$\neg$***FirstSucceed*** $\rightarrow$ ***TryAgain***

JerseyMikes : It's Jersey Mike's.

FreshlySliced : It's freshly sliced.

$\neg$***FreshlySliced*** $\rightarrow$ $\neg$***JerseyMikes***

Contrapositive?

JerseyMikes : It's Jersey Mike's.

FreshlySliced : It's freshly sliced.

$\neg$***FreshlySliced*** $\rightarrow$ $\neg$***JerseyMikes***

JerseyMikes $\rightarrow$ ***FreshlySliced***

An Important Equivalence

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

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

- Later on, this equivalence will be incredibly useful:

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

The Biconditional Connective

Side Note: Contrapositive

We can use truth tables to demonstrate the equivalence of $p \to q$ and $\neg q \to \neg p$.

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

same :)

The Biconditional Connective

- In our previous lecture, we saw that the statement "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?

Biconditionals

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

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.

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:

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

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?

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

- Operator precedence for propositional logic:

$\neg$

$\land$

$\lor$

$\to$

$\leftrightarrow$

- 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.
  - $\land$ and $\lor$ bind more tightly than $\to$.
- We will commonly write expressions like $p \land q \to r$ 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	Negation
$\neg p$	"not"	`!`	Negation	p
$p \wedge q$	"and"	`&&`	Conjunction	$\neg p \vee \neg q$ $p \to \neg q$
$p \vee q$	"or"	`||`	Disjunction	$\neg p \wedge \neg q$
$\top$	"true"	`true`	Truth	$\bot$
$\bot$	"false"	`false`	Falsity	$\top$
$p \to q$	"implies"	*see PS2!*	Implication	$p \wedge \neg q$
$p \leftrightarrow q$	"if and only if"	*see PS2!*	Biconditional	$p \leftrightarrow \neg q$ $\neg p \leftrightarrow q$

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 1:00PM. You have three "late days" you can use throughout the quarter. Each automagically extends assignment deadlines from Friday at 1:00PM to Saturday at 1:00PM; 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 have started (as of Sunday)! 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. Keith's are in Durand 317. Sean's are in Durand 331-B.
- 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 Sunday, Monday, and Tuesday!

Back to CS103!

Why All This Matters

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: $p \leftrightarrow q$

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 $y \geq 8$”

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

“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 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.

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:

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

Next Time

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