Teaching with Lean

Jeremy Avigad

Department of Philosophy
Department of Mathematical Sciences
Hoskinson Center for Formal Mathematics

Carnegie Mellon University

July 2, 2024

The Lean interactive proof assistant

I'll start with a demonstration of Lean.

You can find a similar one on my web page, under Talks, and run it in your browser.

Notes:

- Put your cursor on any keyword with a sguiggly blue underline to see the response from Lean in the information window to the right.
- Move your cursor through any proof to see the proof state change.
- Hover over identifiers and symbols to see popup documentation.

Teaching with proof assistants

Proof assistants have a lot of potential for teaching.

Interaction provides:

- immediate feedback and positive encouragement
- error messages and correction
- information about the current state of a proof
- means to search, experiment, and explore
- increased student engagement

Proof assistants are also generally useful in academia and industry.

There is helpful information on the teaching page of the Lean community web site.

Teaching with proof assistants

Specific drawbacks:

- There is a steep learning curve to using a proof assistant.
- Syntax is fiddly.
- Error messages are confusing.
- Students need to know names of definitions, theorems, functions, and tactics in the library.

Overarching concerns:

- Teaching the specifics of a proof assistant may distract from the concepts that you want students to learn.
- Engaging with technology may discourage students from reflective thought.

Teaching with proof assistants

Strategies to deal with the complexity:

- Teach very specific tasks.
- Choose exercises very carefully.
- Provide lots of examples.
- Provide interfaces that hide some of the complexity.
- Use AI copilots?

Strategies to address the broader concerns:

- Use a proof assistant to supplement rather than replace other assessments.

Teaching with Lean

Some of my favorite resources:

- The Natural Number Game
- The Set Theory Game
- Mathematics in Lean
- Logic and Mechanized Reasoning
- The Mechanics of Proof
- Verbose Lean
- How to Prove it With Lean

There is also a lot of potential for widgets.

Teaching with Lean

I will talk about three uses of Lean for teaching:

1. teaching formalization of mathematics to students in mathematics and computer science:
   - Avigad and Massot, Mathematics in Lean
2. teaching automated reasoning or formal verification to students in computer science:
   - Avigad, Heule, and Nawrocki, Logic and Mechanized Reasoning
   - Blanchette et al., Logic and Verification
3. teaching mathematical proof to anyone:
   - Avigad, Lewis, and van Doorn, Logic and Proof
   - Macbeth, The Mechanics of Proof
   - Massot, Verbose Lean
   - Velleman, How to Prove it With Lean
   - Hazratpour, Introduction to Proofs

Mathematics in Lean

This is a textbook designed to be read alongside examples and exercises in Lean.

Goal: teach students (of all ages) to formalize mathematics as quickly as possible.

This is joint work with Patrick Massot. It is also commonly used in Lean workshops and tutorials.

Readers generally pull a copy of the repository, but they can also use Codespaces or Gitpod.

Mathematics in Lean

Design decisions:

- Don't worry about theory, logic, foundations, or explaining how Lean works.
- Start with basic skills, and build up gradually.
- Introduce information as it is needed.
- Focus on mathematical examples.
- Build text around exercises.
- Later chapters focus on specific domains.

Early on, we provide readers with the library facts they will need, and we ask them to fill in small inferences.

Gradually, we show them how to find the facts they need, and expect them to become more independent.

Mathematics in Lean

The table of contents mirrors an introductory “concepts” course:

- basics (calculation, using theorems and lemmas)
- logic (quantifiers, proof by cases, negation)
- sets and functions
- elementary number theory
- abstract algebra
- . . . and then, specific branches of mathematics.

Mathematics in Lean

I used it in *Interactive Theorem Proving (21-327)* in the Department of Mathematics at CMU in Fall 2022.

Assessments were based on regular assignments, a midterm project, and a final project.

The course went well; students enjoyed it.

I will do it again this coming fall. There are 35 students enrolled and a waiting list of 17.

Philip Wood taught a similar course at Harvard.

Teaching with Lean

I will talk about three uses of Lean for teaching:

1. teaching formalization of mathematics to students in mathematics and computer science:
   - Avigad and Massot, Mathematics in Lean
2. teaching automated reasoning or formal verification to students in computer science:
   - Avigad, Heule, and Nawrocki, Logic and Mechanized Reasoning
   - Blanchette et al., Logic and Verification
3. teaching mathematical proof to anyone:
   - Avigad, Lewis, and van Doorn, Logic and Proof
   - Macbeth, The Mechanics of Proof
   - Massot, Verbose Lean
   - Velleman, How to Prove it With Lean
   - Hazratpour, Introduction to Proofs

Logic and Mechanized Reasoning

In the spring, Marijn Heule and I taught *Logic and Mechanized Reasoning (15-311)* in the Computer Science Department at Carnegie Mellon. (It was a new variation of a course we taught twice before.)

The course can be used to fill a "logic and languages" elective, and had 56 students.

The prerequisites were a functional programming class and an introduction to mathematical proof.

Logic and Mechanized Reasoning

We used a textbook and course materials we started two years earlier with Wojciech Nawrocki:

- the textbook
- the repository
- the course page

We are still revising the materials.

Most students installed Lean, but they could also use Codespaces or Gitpod.

Logic and Mechanized Reasoning

A notable feature of the course is that there are three parallel strands:

- *Theory:* we explore the syntax and semantics of classical propositional logic and first-order logic
- *Implementation:* we implement basic syntactic operations, semantic evaluation, simple decision procedures
- *Application:* we solve interesting problems with SAT solvers, SMT solvers, and theorem provers.

The course is based on Lean 4, which also serves as a front end to CaDiCaL, Z3 or cvc5, and Vampire.

There were programming assignments on the homework, in addition to pen-and-paper exercises.

Theory

Topics:

- syntax and semantics of propositional logic.
- normal forms (NNF, CNF, DNF)
- the Tseitin transformation (and equivalence vs. equisatisiability).
- decision procedures: DP, DPLL, CDCL
- resolution proof and other proof systems
- syntax and semantics of first-order logic
- unification (most general unifiers, etc.)
- decision procedures for equality (congruence closure)
- decision procedures for linear arithmetic
- skolemization, resolution

Implementation

We show students or have them implement:

- translations and conversion to normal forms
- evaluating formulas wrt the semantics
- the Tseitin transformation
- DP, DPLL
- unification and matching
- congruence closure
- linear arithmetic
- ...

Lean's mechanisms for defining custom syntax are very helpful.

Implementation

Semantics $\mathcal{M} \models_\sigma A$ is defined recursively:

- $\mathcal{M} \models_\sigma t = t'$ if and only if $[\![t]\!]_{\mathcal{M},\sigma} = [\![t']\!]_{\mathcal{M},\sigma}$.
- $\mathcal{M} \models_\sigma R(t_0, \ldots, t_{n-1})$ iff $R^{\mathcal{M}}([\![t_0]\!]_{\mathcal{M},\sigma}, \ldots, [\![t_{n-1}]\!]_{\mathcal{M},\sigma})$.
- $\mathcal{M} \models_\sigma \top$ is always true.
- $\mathcal{M} \models_\sigma \bot$ is always false.
- $\mathcal{M} \models_\sigma A \wedge B$ if and only if $\mathcal{M} \models_\sigma A$ and $\mathcal{M} \models_\sigma B$.
- $\mathcal{M} \models_\sigma A \vee B$ if and only if $\mathcal{M} \models_\sigma A$ or $\mathcal{M} \models_\sigma B$.
- $\mathcal{M} \models_\sigma A \to B$ if and only if $\mathcal{M} \not\models_\sigma A$ or $\mathcal{M} \models_\sigma B$.
- $\mathcal{M} \models_\sigma A \leftrightarrow B$ if and only if $\mathcal{M} \models_\sigma A$ and $\mathcal{M} \models_\sigma B$ either both hold or both don't hold.
- $\mathcal{M} \models_\sigma \exists x\, A$ if and only if for some $a \in |\mathcal{M}|$, $\mathcal{M} \models_{\sigma[x \mapsto a]} A$.
- $\mathcal{M} \models_\sigma \forall x\, A$ if and only if for every $a \in |\mathcal{M}|$, $\mathcal{M} \models_{\sigma[x \mapsto a]} A$.

Implementation

```
def FOForm.eval {α} [Inhabited α] [BEq α]
   (M : FOModel α) (σ : FOAssignment α) : FOForm → Bool
| eq t1 t2 => t1.eval M.fn σ == t2.eval M.fn σ
| rel r ts =>  M.rel r (ts.map $ FOTerm.eval M.fn σ)
| tr => true
| fls => false
| neg A => !(eval M σ A)
| conj A B => (eval M σ A) && (eval M σ B)
| disj A B => (eval M σ A) || (eval M σ B)
| impl A B => !(eval M σ A) || (eval M σ B)
| biImpl A B => (!(eval M σ A) || (eval M σ B)) &&
                (!(eval M σ B) || (eval M σ A))
| ex x A => M.univ.any fun val =>
                    eval M (σ.update x val) A
| all x A => M.univ.all fun val =>
                    eval M (σ.update x val) A
```

Implementation

Substitution for terms is defined recursively:

$$\sigma\, x = \sigma(x)$$
$$\sigma\, f(t_1, \ldots, t_n) = f(\sigma\, t_1, \ldots, \sigma\, t_n)$$

```
partial def subst (σ : FOAssignment FOTerm) :
   FOTerm → FOTerm
| var x   => σ x
| app f l => app f $ l.map (subst σ)
```

Recursion is also great for normal form transformations.

Application

We showed students how to use:

- SAT solvers to solve combinatorial problems
- SMT solvers to solve combinatorial problems and test equivalence
- first-order provers to solve logic problems
- Lean to prove propositional and first-order validities

We use Lean as a convenient front end for the first three.

The theory component helped students understand what the tools were doing, and the implementation helped them understand *how* they were doing it.

Logic and Verification

An introduction to proof and program verification with Lean.

- a textbook by Anne Baanen, Alexander Bentkamp, Jasmin Blanchette, Johannes Hölzl, and Jannis Limperg, The Hitchhiker's Guide to Logical Verification
- a course taught at Vrije Universiteit Amsterdam by Blanchette and colleagues
- a course taught at Brown by Robert Y. Lewis.

Teaching with Lean

I will talk about three uses of Lean for teaching:

1. teaching formalization of mathematics to students in mathematics and computer science:
   - Avigad and Massot, Mathematics in Lean
2. teaching automated reasoning or formal verification to students in computer science:
   - Avigad, Heule, and Nawrocki, Logic and Mechanized Reasoning
   - Blanchette et al., Logic and Verification
3. teaching mathematical proof to anyone:
   - Avigad, Lewis, and van Doorn, Logic and Proof
   - Macbeth, The Mechanics of Proof
   - Massot, Verbose Lean
   - Velleman, How to Prove it With Lean
   - Hazratpour, Introduction to Proofs

Logic and Proof

Logic and Proof (with Robert Lewis and Floris van Doorn)

- audience: freshmen and sophomores from a variety of backgrounds; no prerequisites beyond high-school mathematics
- an introduction to symbolic logic, informal mathematical proof, and formal proof.

Originally written for Lean 3. Joseph Hua has been porting it to Lean 4.

Logic and Proof

Goals of the course:

- Teach students to write ordinary mathematical proofs.
- Teach students how to use symbolic logic (to make assertions, prove assertions, and specify properties).
- Teach students to use Lean, in service to the other two goals.

Students could find the textbook and do exercises online:

https://leanprover.github.io/logic_and_proof/

Logic and Proof

Mathematical topics: sets, relations (order, equivalence relations), functions, induction, combinatorics, probability, elementary analysis (the real numbers and limits), axioms of set theory

Logic topics: propositional logic, natural deduction, first-order logic, truth assignments and models (informally), higher-order quantifiers

Lean exercises: e.g. propositional and first-order logic, set-theoretic identities, showing that the composition of surjective functions is surjective, or proving the commutativity of multiplication by induction.

Each strand was standard. The main novelty was in combining them.

Logic and Proof

Question. Let f be any function from X to Y, and let g be any function from Y to Z. Show that if $g \circ f$ is injective, then f is injective.

Give an example of functions f and g as above, such that $g \circ f$ is injective, but g is not injective.

Student answer (to first part). Assume that $g \circ f$ is injective. Then by definition, for all $a, b \in X$, we have that $(g \circ f)(a) = (g \circ f)(b) \implies a = b$.

Now assume that there exist some $x, y \in X$ such that $f(x) = f(y)$. Then we have $(g \circ f)(x) = (g \circ f)(y)$, which implies $x = y$ by the injectivity of $g \circ f$. So f is injective by definition.

Logic and Proof

```
variable (A B C : Type)
variable (f : A → B) (g : B → C)

example (h : Injective (g ∘ f)) : Injective f := by
 intro (x1 : A) (x2 : A)
 intro (h1 : f x1 = f x2)
 have h2 : g (f x1) = g (f x2) := by
  rw [h1]
 show x1 = x2
 exact h h2

example (h : Surjective (g ∘ f)) : Surjective g := by
 intro (z : C)
 obtain ⟨x, h1 : g (f x) = z⟩ := h z
 show ∃ a, g a = z
 use f x, h1
```

Logic and Proof

Observations:

- Make it clear that there are three distinct languages:
  - ordinary mathematics
  - symbolic logic
  - formal proof languages

  Students will not get them confused.
- The parallel developments seemed to help. Students could "see" an exists elimination or an or elimination in an informal proof.
- Students liked the course. There was no clear favorite among the topics: some liked Lean more than the other parts, some less.

The Mechanics of Proof

This is a textbook written by Heather Macbeth, to introduce students at Fordham University to writing proofs.

Students write both pen-and-paper proofs and Lean proofs of the same exercise (in either order!).

She wrote custom tactics and choose the topics and exercises carefully.

Verbose Proof

Verbose Lean 4 is a project by Patrick Massot.

Students write proofs in a controlled natural language.

The system provides (customized) help and hints.

He has been using the method to teach introductory analysis to students in mathematics and computer scientists for a few years.

A paper on this has been accepted to the next ITP conference.

How to Prove it with Lean

Daniel Velleman's How to Prove it With Lean is another Lean-based introduction to proof.

It is a companion to his popular textbook, *How to Prove It*, but it can also be read independently.

Introduction to Proof with Lean

Sina Hazratpour's Introduction to Proof with Lean has materials online.

He has used them successfully with students at Johns Hopkins.

Teaching Mathematical Proof

I am still looking for the ideal approach to teaching mathematical proof:

- *Logic and Proof* spends too much time on logic (e.g. diagrammatic natural deduction).
- Most courses end up tailoring the mathematics to the proof assistant.
- I am hopeful that students can learn to use formal language and informal mathematical language in parallel, at least for *statements*.
- I am hopeful that we can rely on automation to fill in "straightforward" steps in a mathematical argument (modifying what is "straightforward" in each context).

Final thoughts

The Lean community web pages and Zulip chat are a tremendous resource.

More seasoned Lean users help newcomers, who then pay the favor forward.

If we can figure out how to expand this cycle to mathematics and computer science education more broadly, it can become a powerful force.

Final thoughts

We need more educational research:

- Does learning formal reasoning help students learn informal reasoning?
- Do skills with interactive proof assistants transfer?

Anecdotes and intuitions are often misleading.

Final thoughts

Digital technology creates new access points.

Children can find material online and learn how to do mathematics the same way that children find material online and learn how to code.

But the children who benefit the most are those with parents, teachers, and financial resources to support their exploration.

If we are not careful, the digital divide may exacerbate the divide in mathematics and computer science.

We need to think long and hard about how to ensure that everyone benefits.

Conclusions

Digital proof technology is transformative.

It's like the digitization of language (in email, on the web, in databases) but better: mathematical and computational reasoning are inherently formal.

The interest and enthusiasm among young people is encouraging.

Let's give them opportunities to make the most of the new technologies.

Conclusions

It is nice teaching with Lean.

- Students seem to enjoy it.
- I enjoy it.
- There is a supportive community.