Teaching undergraduate mathematicians and computer scientists how to formalize mathematics

Jeremy Avigad

Department of Philosophy
Department of Mathematical Sciences

Hoskinson Center for Formal Mathematics

Carnegie Mellon University

April 18, 2023

Teaching students to formalize mathematics

Two distinct tasks:

- Teaching students how to use a proof assistant.
- Using a proof assistant to teach students how to do mathematics.

The second task is a lot harder:

- Proof assistants are hard to use.
- The technology can become the main focus, distracting from mathematics.

Using formalization to teach mathematics

Why use proof assistants to introduce students to mathematics?

- Students get immediate feedback as to whether a step is correct.
- Students get feedback as to what objects are in play at any point in a proof, what the goals are, etc.
- The formalism helps clarify the structure of informal mathematics.
- The skills (formal and informal) should reinforce each other.
- Formal technology may be useful to students going on to mathematics.
- Formal technology may be useful to students going on to computer science (e.g. software verification).
- Formalization is enjoyable in its own right.

Teaching students to formalize mathematics

Hats off to Rob, Gihan, Paolo, Athina, Kitty, Gila, Heather, and Patrick for doing the brave, difficult thing.

Kevin, Phillip, and I are addressing an easier audience, i.e. students who already know how to read and write mathematical proofs, and who are specifically choosing to learn how to use proof assistants.

The course

In the fall of 2022, I taught a new course at Carnegie Mellon. The course description is online here.

21-321 Interactive Theorem Proving Fall: 9 units
Computational proof assistants now make it possible to work interactively to write mechanically verified definitions, theorems, and proofs. Important theorems have been formalized in this way, and digital libraries are being developed collaboratively by the mathematical community. Formalization of mathematics also serves as a gateway to the use of new technologies for discovery, such as automated reasoning and machine learning. This course will teach you how to formalize mathematics so that you, too, can contribute to the effort. We will explore a logical framework, dependent type theory, which serves as a practical foundation in a number of proof assistants. Finally, as time allows, we will explore ways of automating various aspects of mathematical reasoning. (Three 50 minute lectures)
Prerequisites: 15-151 or 21-128

The course

“21-321” means that the course targets third-year students. It counts as a mathematics elective.

9 units / three 50-minute lectures is standard. (Students need 360 units over 8 semesters to graduate.)

Students must have one of these prerequisites:

- 21-128 Mathematical Concepts and Proofs (Mathematical Sciences, 12 units)
- 15-151 Mathematical Foundations for Computer Science (Computer Science, 12 units)

These are first-year requirements for majors in the two departments.

The course

Wojciech Nawrocki was my teaching assistant.

The goal was simply to teach students to formalize mathematics as quickly as possible.

I had 22 students: 13 from mathematics, 6 from computer science, 2 from ECE, 1 from chemical engineering.

There were 1 freshman, 4 sophomores, 6 juniors, 7 seniors, 3 master's students, and one PhD student.

There were only two women, one transgender student, one member of an underrepresented community.

The assessments

Students were graded on the following:

- weekly assignments (40%)
- a mid-semester project (20%)
- a final project (40%)

There were no exams.

The motivations

Why teach students in mathematics and computer science how to use proof assistants?

- Formalizing something is a good way to understand it better.
- Formal technology may be useful to students going on to do mathematics.
- Formal technology may be useful to students going on to do computer science (e.g. software verification).
- Formalization is enjoyable in its own right.

The textbooks

For the first half of the course, we used Mathematics in Lean.

I spent 2–3 lectures telling them about dependent type theory, for which Theorem Proving in Lean is a good reference.

Toward the end of the course, they were working on projects, and I targeted specific tasks that came up.

The lectures

I lectured using a laptop and projecting on a screen, rather than writing on the blackboard.

That worked pretty well. After the lecture, they had a copy of the Lean file I used.

In a questionnaire at the end of the course, a student pointed out it would be helpful to have a copy of the lecture before class.

Largely because of Covid, I also streamed lectures on Zoom and recorded them, but I'll probably dispense with that in the future.

The schedule

By week:

1. (8/28) course overview and introduction, setting up and using Github, calculational proofs
2. (9/5) Monday holiday; calculation, apply (Chapter 2)
3. (9/12) assignment 2 due; finding theorems, more basics, logic (Chapter 3)
4. (9/19) assignment 3 due; logic, quantifiers, tauto, simp, sets (Chapter 4)
5. (9/26) assignment 4 due; sets, functions, relations, proof by induction (Chapter 5)
6. (10/3) assignment 5 due; induction, algebra (Chapter 6)
7. (10/10) assignment 6 due; finish algebra, lists and inductive types
8. (10/17) Fall break

The schedule

9. (10/24) work on first project; dependent type theory (TPIL), history of logic
10. (10/31) first project due; dependent type theory, if / then, misc topics, style guidelines
11. (11/7) assignment 7 due; inductive types, quotients, finite sums and products; guest lectures from Heather! (polyrith, sheaves, the dihedral group)
12. (11/14) work on second project; comments on first project, misc topics
13. (11/21) work on second project; misc topics, Thanksgiving
14. (11/28) second project due; well founded recursion, filters, storytelling

The repositories

Students had to learn how to use Github.

Lectures (Lean files) and assignments were posted in a private repository that all the students had access to.

They also had to create a private repository for their assignments and share it with us.

Wojciech and I used a Python script to pull the repositories for grading.

Wojciech made a copy of the assignment, added comments and a grade, and pushed.

The projects

The projects were entirely open ended: “formalize something.”

I told students that could work in groups, but they didn’t.

Wojciech and I provided suggestions, guidance, and feedback.

There was a preference for discrete mathematics and analysis.

Toward the end of the semester, we dropped down to two lectures a week, and scheduled extra office hours to work on projects.

The projects

Examples:

- fibonacci numbers
- elementary number theory
- binomial coefficients
- The Lovász local lemma (two students tried it; one finished)
- Bolzano–Weierstrass
- club sets (set theory)
- separation logic
- progress, type preservation, and normalization of the simply type lambda calculus

The projects

A homework problem from another class:

> *Let K be a compact metric space. Suppose the set of functions $\{f_i : K \to \mathbb{R} \mid i \in \mathbb{N}\}$ is uniformly equicontinuous, and suppose $f : K \to \mathbb{R}$ is such that $f_i \to f$ pointwise. Prove that $f_i \to f$ uniformly.*

The projects

A project in descriptive set theory:

```
def schroeder_bernstein
   [measurable_space α] [measurable_space β]
   {f : α → β} {g : β → α}
   (hf : measurable_embedding f)
   (hg : measurable_embedding g) :
  α ≃ᵐ β

def borel_schroeder_bernstein
   [topological_space α] [polish_space α] [borel_space α]
   [topological_space β] [polish_space β] [borel_space β]
   {f : α → β} {g : β → α}
   (fmeas : measurable f) (finj : function.injective f)
   (gmeas : measurable g) (ginj : function.injective g) :
  α ≃ᵐ β
```

The projects

Mathematics competition problems:

- Let $f : \mathbb{R} \to \mathbb{R}$ be a continuous function such that $f \circ f \circ f \equiv id$. Then $f \equiv id$.
- Find all functions f, g, h over the positive integers where f is injective and $f(n)^3 + g(n)^3 + h(n)^3 = 3ng(n)h(n)$.
- Find all functions $f : \mathbb{R} \to \mathbb{R}$ satisfying $f(f(x)^2 + f(y)) = xf(x) + y$.

Here's another one:

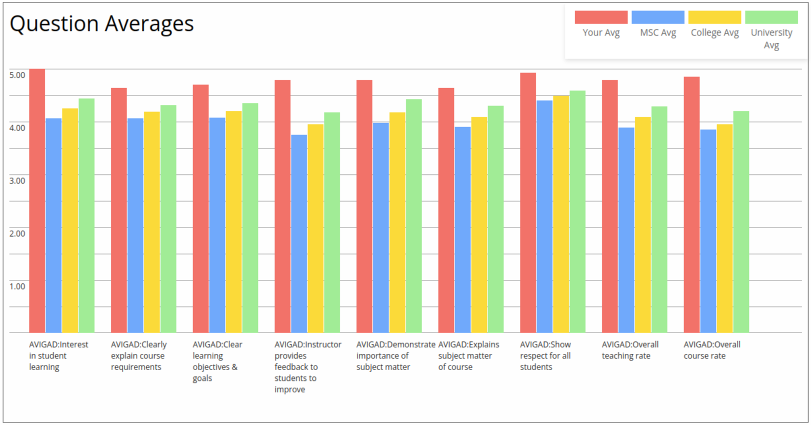
```
theorem sum_mul_exp
  (f : ℝ → ℝ) (h : continuous f)
  (hf : ∀ x y : ℝ, f(x + y) = f x * f y)
  (nzf : f ≠ 0) :
∀ z : ℝ, f z = (f 1) ^ z
```

Zulip

Students were encouraged to introduce themselves on the Lean Zulip channel and ask questions. Many did.

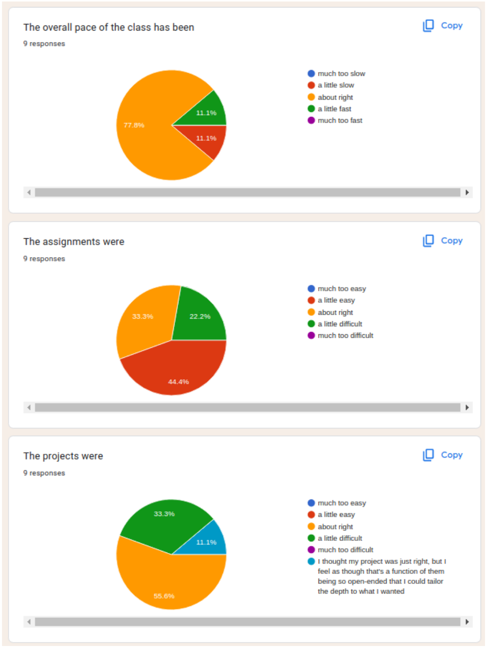
Reception

Students liked the course.



(14/22 responding.)

Reception


The overall pace of the class has been

9 responses

- ● much too slow
- ● a little slow
- ● about right
- ● a little fast
- ● much too fast

The assignments were

9 responses

- ● much too easy
- ● a little easy
- ● about right
- ● a little difficult
- ● much too difficult

The projects were

9 responses

- ● much too easy
- ● a little easy
- ● about right
- ● a little difficult
- ● much too difficult
- ● I thought my project was just right, but I feel as though that's a function of them being so open-ended that I could tailor the depth to what I wanted

Reception

“Thanks so much for this course!!! I had a really great time and will be quick to recommend it! It’s really been such a consistent thing to look forward to every week!”

“It got me excited for the future of this field in mathematics.”

“One of the coolest courses I’ve taken at CMU so far.”

“Great course, make it a requirement for the math major rather than just an elective in the future.”

Reception

“Despite taking this course as a last-minute decision during scheduling last year, it quickly became one of my favorite math courses I’ve taken to date. The material is very approachable and honestly just fun; Lean is a very enjoyable theorem prover to work in, and few things are as satisfying as that ‘Goals Complete!’ at the end of a long proof. The homeworks were always of reasonable difficulty and never took too long to complete, and the open-ended projects allowed me to investigate a particular area of interest (and really reinforce that area through formalization!). All in all, this is a fantastic class, and for other students’ sake I hope it gets offered again soon.”

Conclusions

Teaching students how to formalize mathematics is fun and rewarding.

Having formal skills may be useful to students, and it doesn't seem to hurt.

More importantly, students enjoy it.

Just do it.