Using Lean to Teach Proof Writing

Suhas Alladaboina [1] Sydney Badescu [2] Rohan Bafna [3]

Advisor: Pavel Kovalev [4]

[1]IIT Bombay [2]UC San Diego [3]Georgia Tech [4]Carnegie Mellon

Introduction to Lean

Lean is a *proof assistant*, a piece of software which

- enables mathematicians to check the correctness of mathematical proofs;
- guides mathematicians by keeping track of the progress of a proof as it is being written.

Lean is based on a foundation of mathematics called dependent type theory. This foundation is well suited for proof assistants as it uses a simple first-order logic, which allows computers to easily check the correctness of proofs.

Writing Formal Proofs in Lean

In Lean, a formal proof is written as a statement to prove, followed by a sequence of commands called tactics. Each tactic updates the proof state, which is shown interactively to the user.

Compared to informal proofs, formal proofs:

- use Lean's syntax as opposed to natural language, which makes them easier to parse for computers but harder to understand for humans;
- must be precise and correct—Lean will reject a proof that skips steps or makes mistakes;
- can use advanced features of Lean including proof automation, where Lean automatically tries to look for a proof of a given statement, or can be written manually.

Example: Transitivity of Divisibility

The following proof, written in Lean 4 Web, shows transitivity of divisibility on the integers.

```
1 import Mathlib
2 example (a b c : Int) : c ∣ b → b ∣ a → c ∣ a := by
3   intro c_div_b
4   intro b_div_a
5   rcases c_div_b with ⟨q, hb⟩
6   rcases b_div_a with ⟨r, ha⟩
7   rw [hb] at ha
8   use r * q
9   rw [ha]
10  ring
```

▼ MathlibDemo.lean:6:29

▼ Tactic state

1 goal

▼ case intro.intro

```
a b c q : ℤ
hb : b = c * q
r : ℤ
ha : a = b * r
⊢ c ∣ a
```

- The top window shows the sequence of tactics in the proof.
- The bottom window is the proof state at the stage after line 6.
- At this point in the proof, Lean displays to the user what is known ($b = cq$ and $a = br$) and what still needs to be proven ($c \mid a$).

Objective

We aim to make materials for introductory proof-writing courses that enable professors to integrate Lean into their classes.

- Our content is based off of the Carnegie Mellon courses Concepts of Mathematics (21-127 and 21-128).
- We want to flatten the learning curve for Lean so that students don't feel frustrated and can make good use of Lean's capabilities.
- Learning to write proofs by hand is still important. Our materials supplement, not replace, pen-and-paper proofs.

Benefits of Using Lean

- Lean automatically checks the correctness of proofs, offering instant feedback to students.
- Lean tracks the proof state for students as they write proofs.
- Lean forces students to be rigorous, as students cannot implicitly use facts without stating them.
- Lean is a professional tool, and using it in proof writing course is an easy way to introduce students to it.

Prior Work

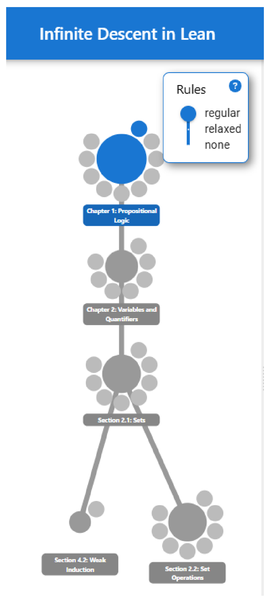
***How To Prove It with Lean* by Daniel J. Velleman; *Mechanics of Proof* by Heather Macbeth**

Online textbooks which introduce students to proof writing using Lean; not ideal for us as we want to flatten Lean's learning curve.

The Natural Number Game

An interactive website that teaches students Lean by having them prove basic properties about the natural numbers. Open sourced, allowing us to write our own similar game.

Our Approach



We are creating a game similar to the Natural Number Game with content appropriate for an introductory proof writing course.

- Content is based on the proof-writing textbook *An Infinite Descent into Pure Mathematics* by Clive Newstead.
- Composed of exercises which teach students proof-writing strategies, based on the exercises in *Infinite Descent*.
- Uses a custom set of tactics, which we try to design so that proofs in the game correspond closely to informal proofs, so that students can practice translating between the two.

The Game in Action

Below is a screenshot from the level that teaches students how to use unique existentially quantified statements in their proofs.

Active Goal

Objects:

n : ℤ

Assumptions:

h : ∃! x, n * x = 0

Goal:

n ≠ 0

exists_unique_elim h into x, hexists, hunique | Retry

Our goal is a negated statement, so use the `by_contra` tactic to prove it using proof by contradiction.

Active Goal

Objects:

n : ℤ

x : ℤ

Assumptions:

h : ∃! x, n * x = 0

hexists : n * x = 0

hunique : ∀ (y : ℤ), n * y = 0 → y = x

Goal:

n ≠ 0

Execute

Breakdown of its parts:

- Above the light blue box, the proof state is shown for the current stage of the proof.
- The light blue box contains a hint after the student correctly progresses the proof using the `exists_unique_elim` tactic.
- After the light blue box, the game updates the proof state with the new hypotheses `hexists` and `hunique`.

Here is an example where a student makes a mistake, by trying to prove a disjunction with the `and_intro` tactic, which is used to prove conjunctions. Our game reports the error to the student and lets them retry the tactic.

Active Goal

Goal:

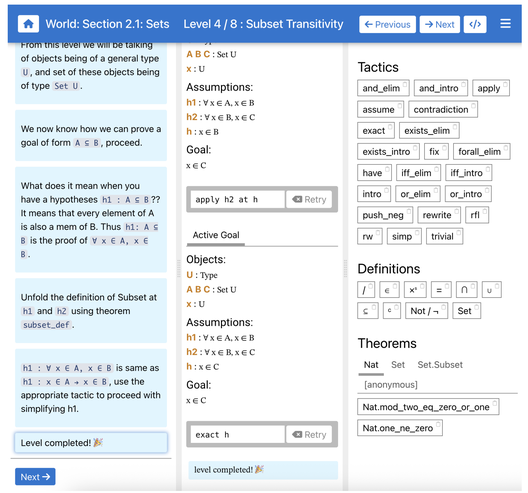
2 + 2 = 4 ∨ 3 < 2

Failed command: and_intro

tactic 'and_intro' failed, the goal 2 + 2 = 4 ∨ 3 < 2 isn't a conjunction
⊢ 2 + 2 = 4 ∨ 3 < 2

and_intro | Execute

A Full Level



Tactics

Tactics are the individual steps of a proof in Lean. Lean provides its own suite of tactics, but also allows users to write their own. We deemed Lean's default tactics too powerful and complicated to be suitable for students, so we designed a suite of alternative tactics for our game.

Compared to Lean's default tactics, ours:

- are more specialized to particular situations
- have more predictable behavior
- employ a consistent naming convention

Challenges & Considerations

Using Lean to teach proof-writing instead of teaching Lean itself

Because we do not teach students how to use Lean on their own, they will not be as equipped to formalize their own proofs outside of our game.

Level of detail between informal and formal proofs

Formal proofs require students to explicitly invoke lemmas and basic facts which students may take for granted in an informal proof. This is good for rigor, but can make proofs in Lean tedious to write without automation. Writing tactics that have the right level of detail and finding exercises that have the right level of difficulty is a delicate balance.

Lean helps with rigor, but not intuition

Lean forces students to write proofs that follow strict rules of logic, but it does not help students gain intuition about mathematics as well as writing proofs on pen and paper can.

Next Steps

Currently, we are finalizing the levels for chapters 1 through 3 (elementary logic, sets, and functions) of *Infinite Descent* and adapting chapters 4 through 6 (induction, relations, and infinite sets). After that, we hope to release our project publically.