Teaching with Computer-Based Proof Assistants: Perspectives from Instructors of Mathematics

Education Notes June 2022 (Vol. 54, No. 3)

Gila Hanna (OISE, University of Toronto)

Xiaoheng Yan (OISE, University of Toronto)

Education Notes bring mathematical and educational ideas forth to the CMS readership in a manner that promotes discussion of relevant topics including research, activities, issues, and noteworthy news items. Comments, suggestions, and submissions are welcome.

John McLoughlin, *University of New Brunswick (johngm@unb.ca)*
Kseniya Garaschuk, *University of Fraser Valley (kseniya.garaschuk@ufv.ca)*

The authors are engaged in a project to investigate the potential benefits of teaching proof to undergraduate students with the help of a computer-based proof assistant. A point of departure was the observation that such proof assistants do not play a role in the undergraduate mathematics curriculum in North America that is in any way commensurate with their increasingly important role in mathematical practice. Our hypothesis is that a proof assistant could be a useful additional tool in teaching undergraduate mathematics, with the potential to foster mathematical understanding and in particular students' competence in proving (Avigad, 2019; Buzzard, 2020; Hanna & Yan, 2021; Thoma & Iannone, 2021). In preparation for our study, we had looked at a number of proof assistants, such as Coq, HOL, and Lean, recently acknowledged as having an important role in mathematics research (Castelvecchi, 2021).[1]

Here we report that there are a number of educators who have already chosen to include the use of a proof assistant in their mathematics courses. We contacted nine of them, to ask for any insights and suggestions stemming from their teaching experience (see Table 1). In what follows, we set out a summary of their responses to our five questions about teaching with the help of a computer-based proof assistant. The specific proof assistant these teachers had all been using is Lean, an interactive theorem prover (ITP) that provides instantaneous feedback, and allows exploration. Lean has a very active and friendly community of users who focus on mathematics rather than on computer science and software engineering. (See *Notes* following the references for further insight and relevant resources concerning Lean.)

Table 1 Participants

Country	Number of Participants
France	1
The Netherlands	1
United Kingdom	1
United States	6
Total	9

Instructors' Responses

1. Was the course you taught using an interactive theorem prover (ITP) an introduction to mathematical reasoning? To logic in general? To proof? Other?

Interactive theorem provers (ITPs) have been used in the following mathematics courses:

- Introduction to Logic for 1st-year students of computer science
- Undergraduate Introduction to Mathematical Reasoning
- Undergraduate Introduction to Proofs
- Undergraduate Real Analysis
- Undergraduate Computer-assisted Logic and Proofs (50 students enrolled in the first year, all pursuing a double major in mathematics and computer science)
- Graduate Introduction to Logic

Most instructors reported that they used the ITP, Lean, in introductory courses oriented towards logic and mathematical reasoning. It is important to note that Lean includes resources for several other topics in mathematics, such as Analysis, Linear algebra, and Geometry.[2]

2. How did you incorporate an ITP into your teaching?

Interactive theorem provers were incorporated into teaching mainly in three ways:

- *to demonstrate proofs in* a lecture (5 out of 9 instructors)
- *for in-class problem-solving activities* (all instructors)
- *for* homework/assignment (4 out of 9 instructors)

The mathematics instructors reported devoting a portion of their class time to demonstrating how Lean could be used to formalize proofs of theorems that had been proven less formally in their lectures. The amount of time spent on Lean depended on the length of the course as a whole.

For example, one instructor dedicated 5 out of 35 class periods to the Lean interactive theorem prover. In those 5 class periods, about one-half of the time was devoted to Lean demonstrations, in which new concepts and worked examples were introduced. The other half of the time was spent on group problem-solving in Lean. In addition, there were 5 homework assignments involving Lean following up on the lectures. Lean, however, was not included in any of the exams.

Similarly, another instructor spent about half an hour in class on Lean demonstrations. Occasionally, she would post Lean formalizations of theorems from the course on her course website. She would also include in each homework assignment one Lean problem for extra credit points. The points were allocated such that the Lean problem could replace any other problem on the homework. To assist students in using Lean, she provided links to online tutorials and support during office hours, but she did not teach the students to use the software in any other way.

The instructor who taught the undergraduate Computer-assisted Logic and Proofs course intentionally provided lecture notes using the Lean format and gave students access to a Lean program he had designed for his class. At the end of the course, students were examined on Lean proofs as well as on paper proofs.

3. Could you describe the learning curve of the ITP for you as an instructor and for your students?

Instructors who had already been using various proof assistants reported the learning curve as relatively flat. Top students and those who had previous programming experience seemed to learn Lean effortlessly. In fact, for many students, learning Lean was very helpful; for others, Lean did not “click” even by the end of the course. For both instructors and students without prior programming experience, the learning curve was steep in the beginning.

One instructor pointed out that it is not easy to assess the Lean learning curve, because it is hard to distinguish difficulties that stem from Lean from those which flow from the mathematical content. The main barriers, according to the instructor, seemed to be issues installing Lean and misunderstandings in reading the Lean user interface. Some instructors suggested that, with more careful attention and support in the first few weeks of the course, it would be possible to increase the number of students who enjoy the Lean part of the course.

4. *From your observation, what worked well when using an ITP in the classroom? What were the major challenges?*

For some students, using Lean made the material less “dry”, and motivated them. One instructor described the outcome of his course using the ITP as follows: Good students really learn a lot, and some average students are pleased too, but weak ones don’t get anything out of it, even when they get decent grades. As other instructors put it, an ITP works best with mathematically mature students.

The most obvious benefit of using an ITP is that students get a much better idea of what a proof really is, because setting out a proof in Lean forces them to organize their thoughts. Another potential benefit of an ITP is in performance assessment. One instructor mentioned that if all calculus homework were turned in to the instructor in Lean format, then the instances of search-copy-paste solutions to assignments would approach zero.

However, students face some obvious challenges in learning and using the ITP:

- The necessarily rigid ITP syntax can be frustrating
- Learning an additional language (i.e., Lean syntax) is a hurdle for some students
- *Compared to a handwritten proof, completing a proof in Lean may require multiple attempts*

5. *In your view, which group of students at the post-secondary level would benefit most from using LEAN in the classroom?*

The short answer to this question is “all of them!”, to quote the very phrasing used by many of the instructors. The instructors were of one mind on this, however they may have expressed it. Their answers to this question may seem incompatible with the ones they gave when asked about the major challenges they faced (Question 4). One respondent did say that weak students do not seem to “get anything out of it”, and another that it “works best with mathematically mature students”. But all agreed that undergraduate students stand to benefit from using Lean, albeit to varying degrees. The two groups mentioned as most likely to make effective use of Lean were students of the philosophy of mathematics and students who double-major in mathematics and computer science.

Final Thoughts

Exploring the perspectives of instructors who used Lean in their teaching provided some indication of both benefits and challenges. We also noted that all the instructors plan to continue using Lean in their courses, despite the challenges it presents. We intend to carry out a systematic evaluation of the degree to which the ITP Lean could be a useful additional tool in teaching undergraduate mathematics. Such an evaluation could help in the design of new and effective teaching strategies that can make ITPs valuable in an educational setting.

References

Avigad, J. (2019) Learning logic and proof with an interactive theorem prover. In Hanna, G., Reid, D. & de Villiers, M. (Eds.), *Proof technology in mathematics research and teaching* (pp. 277-290). Springer.

Buzzard, K. (2020). Proving theorems with computers. *Notices of the American Mathematical Society, 67*(11), 1791-1799.

Castelvecchi, D. (2021). Mathematicians welcome computer-assisted proof in ‘grand unification’ theory. *Nature, 559*, 18-19. https://www.nature.com/articles/d41586-021-01627-2

Hanna, G., & Yan, X. (2021). Top of Form

Bottom of Form

Opening a discussion on teaching proof with automated theorem provers. *For the Learning of Mathematics, 41*(3), 42-46.

Thoma, A., & Iannone, P. (2021). Learning about proof with the theorem prover LEAN: the

Abundant Numbers Task. *International Journal of Research in Undergraduate Mathematics Education*. https://doi.org/10.1007/s40753-021-00140-1

***Notes*:**

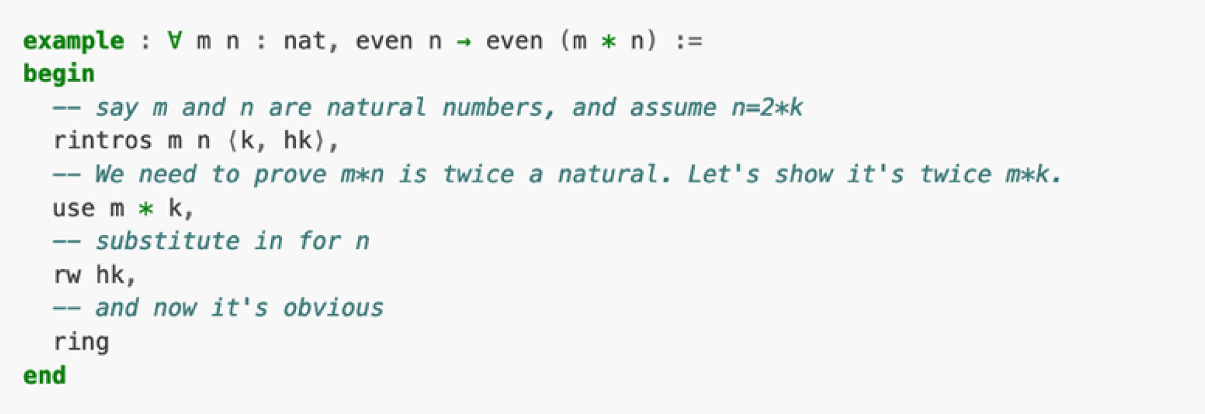
The Lean theorem prover is a proof assistant developed principally by Leonardo de Moura at Microsoft Research.

The Lean mathematical library, *mathlib*, is a community-driven effort to build a unified library of mathematics formalized in the Lean proof assistant. The library also contains definitions useful for programming. This project is very active, with many regular contributors and daily activity.

The contents, design, and community organization of mathlib are described in the paper The Lean mathematical library, which appeared at CPP 2020. You can get a bird’s eye view of what is in the library by reading the library overview. You can also have a look at our repository statistics to see how it grows and who contributes to it.

Example 1: A proof of if n is even then so is m * n:

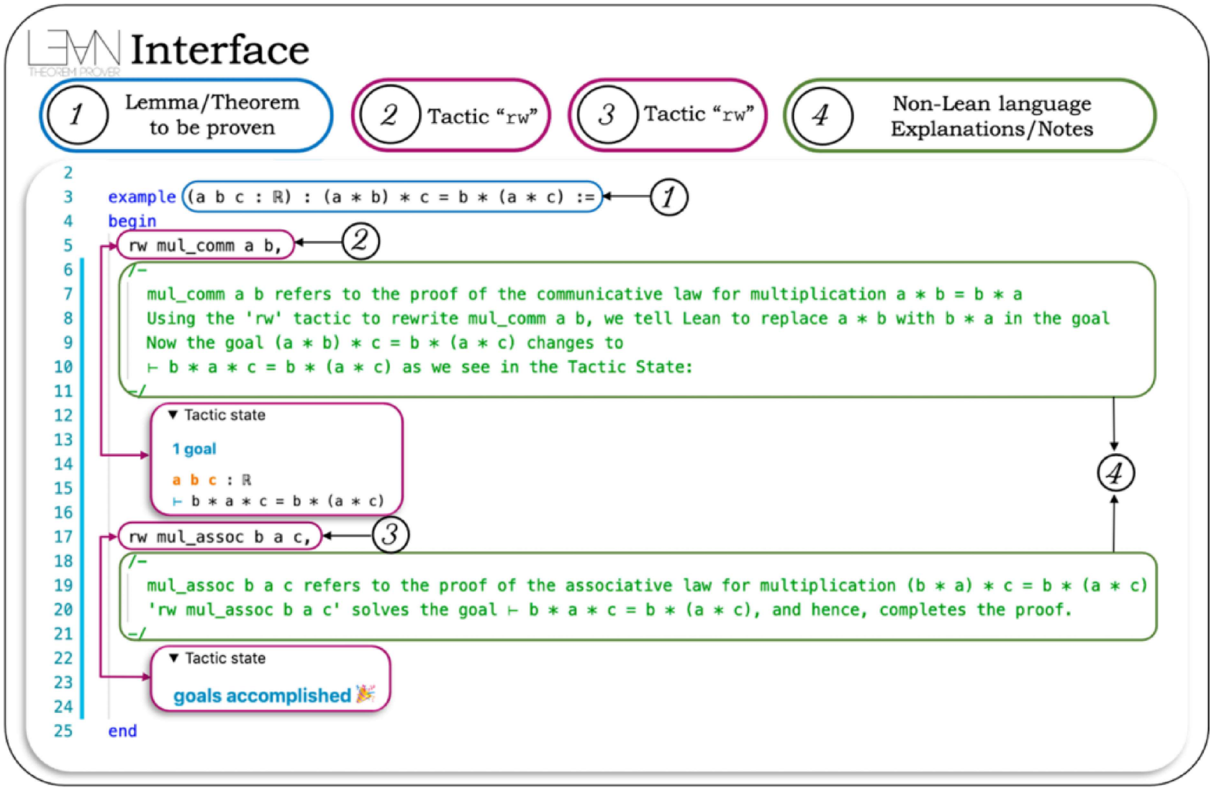
From: https://leanprover-community.github.io/mathematics_in_lean/01_Introduction.html#overview



```
example : ∀ m n : nat, even n → even (m * n) :=
begin
  -- say m and n are natural numbers, and assume n=2*k
  rintros m n ⟨k, hk⟩,
  -- We need to prove m*n is twice a natural. Let's show it's twice m*k.
  use m * k,
  -- substitute in for n
  rw hk,
  -- and now it's obvious
  ring
end
```

As you enter each line of such a proof in VS Code, Lean displays the *proof state* in a separate window, telling you what facts you have already established and what tasks remain to prove your theorem. You can replay the proof by stepping through the lines, since Lean will continue to show you the state of the proof at the point where the cursor is. In this example, you will then see that the first line of the proof introduces m and n (we could have renamed them at that point, if we wanted to), and also decomposes the hypothesis even n to a k and the assumption that n = 2 * k. The second line, use m * k, declares that we are going to show that m * n is even by showing m * n = 2 * (m * k). The next line uses the rewrite tactic to replace n by 2 * k in the goal, and the *ring* tactic solves the resulting goal m * (2 * k) = 2 * (m * k).

Example 2: A proof of (*a* x *b*) x *c* = *b* x (*a* x *c*)


LEAN THEOREM PROVER Interface
1 Lemma/Theorem to be proven
2 Tactic "rw"
3 Tactic "rw"
4 Non-Lean language Explanations/Notes
1
2
3
4

```
example (a b c : ℝ) : (a * b) * c = b * (a * c) :=
begin
  rw mul_comm a b,
  /-
    mul_comm a b refers to the proof of the communicative law for multiplication a * b = b * a
    Using the 'rw' tactic to rewrite mul_comm a b, we tell Lean to replace a * b with b * a in the goal
    Now the goal (a * b) * c = b * (a * c) changes to
    ⊢ b * a * c = b * (a * c) as we see in the Tactic State:
  -/
```

```
▼ Tactic state

1 goal

a b c : ℝ
⊢ b * a * c = b * (a * c)
```

```
  rw mul_assoc b a c,
  /-
    mul_assoc b a c refers to the proof of the associative law for multiplication (b * a) * c = b * (a * c)
    'rw mul_assoc b a c' solves the goal ⊢ b * a * c = b * (a * c), and hence, completes the proof.
  -/
```

```
▼ Tactic state

goals accomplished 🎉
```

```
end
```

The diagram above brings together what's happening in the Lean interface when one works on a simple proof such as (*a* x *b*) x *c* = *b* x (*a* x *c*), marked as 1. To prove it, using the "rw" tactic with the commutative law and associative law for multiplication closes the goal. The commands in Lean language are marked as 1, 2, and 3. Commands 2 and 3 lead to changes in the tactic state where Lean shows the current goal(s) of the proof. The green text, marked as 4, denotes informal math language that could be inserted between the tactics to communicate with the reader about the proof. This space is particularly useful to instructors who want to add explanations or share notes with students.