

# Using the Proof Assistant Lean in Undergraduate Mathematics Classrooms

Gila Hanna<sup>1</sup>, Brendan Larvor<sup>2</sup>, Xiaoheng (Kitty) Yan<sup>1</sup>

<sup>1</sup> The University of Toronto, Canada <sup>2</sup> The University of Hertfordshire, UK

## Abstract

In this paper we develop a case for introducing a new teaching tool to undergraduate mathematics. Lean is an interactive theorem prover that instantly checks the correctness of every step and provides immediate feedback. Teaching with Lean might present a challenge, in that students must write their proofs in a formal way using a specific syntax. Accordingly, this paper addresses the issue of formalism from both a theoretical and a practical point of view. First, we examine the nature of proof, referring to historical and contemporary debates on formalization, and then show that in mathematical practice there is a growing rapprochement between strictly formal proof and proofs-in-practice. Next, we look at selections from the mathematics education literature that discuss how and when students advance through higher levels of mathematical maturity to reach a point at which they can cope with the demands of rigorous formalism. To probe the integration of Lean into teaching from an empirical point of view, we conducted an exploratory study that investigated how three undergraduate students approached the proof of double negation with Lean. The findings suggest that the rigorous nature of Lean is not an obstacle for students and does not stifle students' creativity in writing proofs. On the contrary, proving with Lean offers a great deal of flexibility, allowing students to follow different paths to creating a valid proof.

**Key words:** Formalization of mathematics; Proof assistant, Lean prover; Proof construction; Rigorous and formal proofs; Undergraduate mathematics.

Students taking university-level mathematics courses are expected to master the concept of mathematical proof. This presents a challenge to students and their teachers. Taking into account the significant renewal of interest in the formalization of mathematics (Avigad, 2023) as well as the increased use of computer proof assistants (Buzzard, 2020; Castelvechhi, 2021), we discuss compelling conceptual and cognitive grounds for increased attention to formal proofs in the undergraduate mathematics curriculum, with the aid of computer proof assistants. We first consider conceptual aspects by reviewing two ideal conceptions of proof: 'Cartesian' and 'Leibnizian', as well as the work of researchers in formalization which has shown that the intersection between strictly formal proofs and proofs-in-practice, while small, is growing. Next, we use mathematics education literature to identify the cognitive demands of axiomatic mathematics and the higher levels of mathematical maturity required. With these issues in mind, we then report on an exploratory study that examined how three undergraduate students used the proof assistant Lean to approach the proof of double negation.

## 1. Conceptual aspects: There is no unproblematic definition of mathematical proof

The late Ian Hacking, in his final *Why is there Philosophy of Mathematics at all?* (2010) distinguished two ideal conceptions of proof: ‘Cartesian’ and ‘Leibnizian’. These are ideal in something like the sense of Max Weber’s ideal types, that is, they are distillations of recognisable phenomena. However, they are also ideals in the normative sense that they describe desirable features of proofs that mathematicians might want to strive for. ‘Cartesian’ proofs are the sort that one can (with enough study) come to understand in a single mental act. They are short and they are satisfying, perhaps because they supply the reader with an ‘aha!’ feeling, or a sense of understanding why the result holds, or are especially elegant. The proofs that we usually use to illustrate mathematical thinking tend to be of this sort, such as the mutilated chequerboard proof. Imagine a standard eight-by-eight chequerboard or chessboard with two diagonally opposite corner squares cut out. Let there also be a set of dominoes with each domino the perfect size and shape to cover two squares on the chequerboard. Is it possible to arrange dominoes so as to cover all the uncut squares on the chequerboard with no overlapping? One might try various arrangements and hope to stumble on the answer. However, there is a neat shortcut. From the fact that each domino covers one white and one black square, it’s easy to see that any arrangement of dominoes must cover equal numbers of black and white squares, and then it is obvious that the answer is ‘no’, because the mutilation removed two squares of the same colour. Once you see this point, you can see the whole proof, and you can enjoy having cut through the complex mess of possible domino tilings with a single, simple insight<sup>1</sup>. The proofs in Aigner and Ziegler’s *Proofs from THE BOOK* (2010) aim at this ‘Cartesian’ quality. This collection was inspired by Paul Erdős’s imagined book of perfect proofs maintained by God. The proofs in Aigner and Ziegler’s version of *The Book* are all short (most of them are less than one page in length) and rely on one mathematically satisfying insight. For his part, Hacking offered a proof by Littlewood (1953) as an example of the ‘Cartesian’ ideal type:

A square can be dissected into finitely many unequal squares, but a cube cannot be dissected into finitely many unequal cubes. *Proof of the latter:*

In a square dissection the smallest square is not at an edge (for obvious reasons). Suppose now a cube dissection does exist. The cubes standing on the bottom face induce a square dissection on that face, and the smallest of the cubes on that face stands on an internal square. The top face of this cube is enclosed by walls; cubes must stand on this top face; take the smallest – the process continues indefinitely.

---

<sup>1</sup> Grenier & Payan (2006) explore the classroom use of generalisations of the chequerboard problem, namely whether a shape composed of squares of a given size can tile a larger shape composed of squares the same size, under various constraints. One might, for example, ask students to explore whether it’s possible to tile a zigurat with an L-shape rather than a domino. Where a tiling is provably impossible, will the strategy of the original proof always work? Alekhovich (2004) discussed the challenge that the chequerboard presents to automated reasoning systems. Tanswell (2015) argues that the various formalizations of the basic chequerboard proof are sufficiently different from each other that we cannot speak of *the* formalization of an informal proof.

What makes such proofs ‘Cartesian’ is that they depend on our having clear and distinct ideas about the subject matter (Hacking does not put the point quite in these Cartesian terms). These proofs display the power of mathematical ideas and mathematical thought. They supply the sort of intellectual satisfaction that often draws people to mathematics and sustains their interest. For Erdős, for Aigner and Ziegler, and perhaps for mathematicians generally, proofs of this sort are desirable. They imitate God’s own proofs.<sup>2</sup>

In Hacking’s other ideal type of proof, the so-called ‘Leibnizian’<sup>3</sup>, “every step is meticulously laid out, and can be checked, line by line, in a mechanical way” (p. 21). Proofs of this sort do not require the reader to have any ideas (even unclear and indistinct ones) about the subject matter, because the inferences are applications of content-independent logical rules. They can be of any length because they do not require the reader to gather the whole argument into a single Gestalt. Above all, they can be checked by machines which, while not infallible, are not subject to fatigue, confusion, or motivated reasoning. In Leibniz’s day, automated proof-checking was a dream. Today, we have computer systems that can help mathematicians to build and check proofs, provided that the proofs are written in a machine-appropriate form. Perhaps in time, such systems can take up some of the refereeing burden on mathematics journals (this was the hope of Voevodsky (2014), who suggested that in the future, journals will only accept papers that come with a certificate of machine verification). Perhaps they can be used to check the validity of proofs where the mathematics is only understood by the team creating the proof (this is the mathematical version of the replication crisis) or is simply too voluminous for one person to understand all the ideas. So far, the experience is that preparing a proof for mechanical checking is a major task that requires operators who understand both the checking system and the mathematics in its smallest details. If the refereeing problem is due to the scarcity of such experts, it will not be solved by the current generation of proof assistants. The best documented case of using a theorem prover to assist in checking a new proof is Peter Scholze’s appeal to the Lean community for help with the Clausen-Scholze proof (the Liquid Tensor Experiment). Scholze found that his back-and-forth with the ‘Leaners’ deepened his understanding of his own proof (Granville, forthcoming). The use of Lean did not remove the need for human reviewers, but it did help humans to help another human do mathematics.

Hacking’s ‘Leibnizian’ ideal is recognisable both in the projects to formalize significant pieces of mathematics (including a proof of the independence of the continuum hypothesis from ZFC and the Liquid Tensor Experiment already mentioned<sup>4</sup>), and in generic statements about the nature of proof. Here, for example, is Thomas Hales:

The ultimate standard of proof is a formal proof, which is nothing other than an unbroken chain of logical inferences from an explicit set of axioms. While this may

---

<sup>2</sup> Erdős said that you need not believe in God but, as a mathematician, you should believe in The Book (Aigner and Ziegler 2010, v).

<sup>3</sup> This label is both appropriate (because Leibniz did indeed foresee the potential of formalized mathematics) and misleading (because Leibniz’s philosophy of mathematics was intimately related to his views about logic and metaphysics in ways that are not relevant to the present discussion).

<sup>4</sup> [https://leanprover-community.github.io/lean\\_projects.html](https://leanprover-community.github.io/lean_projects.html)

be the mathematical ideal of proof, actual mathematical practice generally deviates significantly from the ideal. (Hales, 2012, p. x)

Or this, from Joel D. Hamkins:

Proof... lies solidly on the syntactic side, since ideally one can verify and analyze a proof as a purely syntactic object, without a concept of meaning and without ever interpreting the language in any model. (Hamkins, 2020, p. 158)

Hales and Hamkins both seem to use ‘ideal’ to mean something desirable but rarely if ever attained. Formal (‘Leibnizian’) proof is, as Hacking argues, one ideal of proof, and it offers one of the things that mathematicians want from proofs, namely, certainty (which is why Hales offers it as ‘the ultimate standard’). Certainty, however, is not the only benefit of formalization. As Scholze reported, it can also bring clarity, specify precisely the required premises of a proof (so called ‘reverse mathematics’) and facilitate modularity (chunks of code can be re-used more easily if proofs are all written in the same formal language). The cost is the loss of the attractive features of ‘Cartesian’ proofs: brevity, elegance, insight and explanation.

Hacking presented these two sorts of proof as ideal types, and reminded his readers of Wittgenstein’s remark that mathematics is a motley (*ein BUNTES Gemisch*<sup>5</sup>) of techniques of proof (Wittgenstein, 1956, p. 84). The *Gemisch* of proof techniques has only got more *buntes* since Wittgenstein made this observation, since it now includes various kinds of computer-assisted proof (the case-checking procedure that established the four-colour theorem is a different use of computers from systems like Lean, for example) as well as conceptual mathematical techniques that have been developed since then. Along with these developments there is of course a rich philosophical literature on the nature of mathematical proof, from G.H. Hardy’s dismissive claim that,

...there is, strictly, no such thing as mathematical proof; ...we can, in the last analysis, do nothing but point; ...proofs are what Littlewood and I call gas, rhetorical flourishes designed to affect psychology, pictures on the board in the lecture, devices to stimulate the imagination of pupils. (Hardy 1929, p. 18)

Through to reflections on the way in which digital technologies may change what human mathematicians do when they create proofs, such as (Avigad, 2023) and (Granville, forthcoming). Taking the spread of mathematical practice and philosophical discourse (including the philosophical remarks quoted here from Hales, Hamkins and Hardy) together, it is evident that there is no tidy definition of proof that will capture neither what mathematicians do when they prove, nor what they say about proof. There is certainly no tidy definition that can capture both. Notably, the *Princeton Companion to Mathematics* does not

---

<sup>5</sup> “Die Mathematik ist ein BUNTES *Gemisch* von Beweistechniken.” (Capitals and italics in original). Hacking discusses the difficulty of translating ‘BUNTES *Gemisch*’ (Hacking 2014 pp. 57–58). Anscombe’s translation has ‘motley’. A more direct rendering might be a ‘BOUNTEOUS *mixture*’, that is, a mixture of wildly heterogenous items. A miscellany.

offer a definition of proof, preferring, prudently, to give a historical discussion. It ends with another deployment of the useful word ‘ideal’,

The formal notion of proof... continues to provide an ideal model for the principles that underlie what most mathematicians see as the essence of their discipline. It... falls short of explaining the changing ways in which mathematicians decide what kinds of arguments they are willing to accept as legitimate in their actual professional practice. (Gowers, Barrow-Green, & Leader, 2010, p. 142)

Even if we suspect that Wittgenstein overstates the heterogeneity of mathematical proofs, the concept is evidently complex and subject to powerful internal tensions between its Cartesian and Leibnizian ideals.

What does this mean for the use of proof assistants in the teaching and learning of mathematical proof? It seems we lose the benefits of ‘Cartesian’ proofs when we make proofs more formal, and perhaps we do as readers of proofs but not necessarily as makers of proofs, and it is as proof authors that students use systems such as Lean.

## **2. Cognitive aspects: Stepping up to a formal logical structure of proof**

While the concept of mathematical proof in general is complex and perhaps even incoherent, the concept of formal proof is not. Recall that a formal proof is not an argument stated in a natural language, but rather one that uses an unambiguous notation that in principle can be checked mechanically. The universally accepted definition of a formal proof is that it is a finite sequence of sentences, each of which is either an axiom or follows from preceding sentences in the sequence by explicitly stated topic-neutral rules of inference.

Undergraduates are expected to construct proofs that are more complex and more rigorous than those they would have dealt with at their earlier educational levels. Thus, it is to be expected that their transition from secondary to post-secondary mathematics would present cognitive challenges. In the following, we look at selections from the literature that discuss how students advance to higher levels of mathematical maturity. Of particular interest is the process by which students achieve the maturity necessary to understand the demands of rigorous proof.

David Tall is a mathematician and mathematics education scholar who has focused on the cognitive aspects of proof and written extensively on the different levels at which mathematics, and proof in particular, can be understood. Tall (2008) describes the development of a young person’s grasp of mathematics to that of an adult as a passage through three different modes of thinking, which he describes as “three worlds of mathematics”:

- the conceptual-embodied world, based on perception of and reflection on properties of objects, initially seen and sensed in the real world but then imagined in the mind
- the proceptual-symbolic world that grows out of the embodied world through action (such as counting) and is symbolised as thinkable concepts (such as number) that function both as processes to do and concepts to think about (procepts)
- the axiomatic-formal world (based on formal definitions and proof), which reverses the sequence of construction of meaning from definitions based on known objects to formal concepts based on set-theoretic definitions. (p. 7)

Tall used of short terms to represent these three worlds: embodiment, symbolism, and formal. He describes the progress of university mathematics students from intuition to rigour, with the major shift occurring when they move from the embodiment and symbolism of school mathematics to the formalism of advanced mathematical thinking at the university level. Tall has found support for his elaboration of the three worlds of mathematics in empirical studies, such as those of Pinto (1998) and Weber (2004).

Looking at deductive reasoning, Harel and Sowder (1998) developed a cognitive framework that proposes a taxonomy of students' proof schemes. They based their framework on observations and teaching experiments with undergraduate mathematics students. The authors' analysis of proving identifies mental processes such as ascertaining and persuading, the former referring to removing self-doubts and the latter referring to removing other's doubts. They stated that there are three non-mutually exclusive proof schemes held by a prover: 1) external conviction (including acceptance of authority – “my professor said so”); 2) empirical (including inductive evidence and examples); and 3) analytical (where students understand the use of definitions, axioms, and rules of inference). Harel and Sowder see these schemes as representing hierarchical cognitive stages in a student's mathematical development, progressing from the least to the most sophisticated.

Harel (2007) then revised the original proof schemes framework in view of additional empirical data from student interviews and of some historical considerations. He elaborated each scheme and assigned each a new label as follows: “In the new framework they are labelled the external conviction proof scheme class, the empirical proof scheme class, and the deductive proof scheme class” (p. 66). In addition, Harel (2007) refined the three schemes by attaching to them new categories. For example, the deductive proof scheme now consists of two categories, one transformational and the other axiomatic.

Durand-Guerrier, Boero, Douek, Epp, and Tanguay (2012), advanced the view that an emphasis on formal logic in mathematical reasoning and proving is a necessary element in teaching proof. They cite empirical research carried out by Epp (2003) and Durand-Guerrier (2003) which indicates that undergraduate students who had not been taught elementary logic often fail to conclude whether a mathematical statement is true or false. Thus, Durand-Guerrier et al. believe that it is essential for students to have a good level of understanding of the fundamental rules of predicate logic, to be able to construct valid proofs and avoid invalid deductions. To help students improve their proving skills “it is important to view logic as dealing with both the syntactic and the semantic organization of mathematical discourse.” (Durand-Guerrier et al., 2012, p. 385). In their view, undergraduate students do possess the maturity necessary to understand the relevant rules of logic.

We quote here the thoughts of the Fields medalist mathematician Terrence Tao (2009): (points 1, 2, and 3 are a verbatim quotation)

1. The “pre-rigorous” stage, in which mathematics is taught in an informal, intuitive manner, based on examples, fuzzy notions, and hand-waving. The emphasis is more on computation than on theory. This stage generally lasts until the early undergraduate years.
2. The “rigorous” stage, in which one is now taught that in order to do maths “properly”, one needs to work and think in a much more precise and formal manner. The emphasis is now primarily on theory; and one is expected to be able to comfortably manipulate abstract mathematical objects without focusing too much on what such

objects actually “mean”. This stage usually occupies the later undergraduate and early graduate years.

3. The “post-rigorous” stage, in which one has grown comfortable with all the rigorous foundations of one’s chosen field and is now ready to revisit and refine one’s pre-rigorous intuition on the subject, but this time with the intuition solidly buttressed by rigorous theory. The emphasis is now on applications, intuition, and the “big picture”. This stage usually occupies the late graduate years and beyond.

These various schemes do not all coincide. For example, one would expect to find Tall’s proceptual-symbolic mathematics (his second stage) in Tao’s first, pre-rigorous stage. Tall’s sequence ends at his formal level (perhaps because his interest is principally in school mathematics), while Tao (reflecting on the training of research mathematicians) recognises a further stage, in which educated intuition is supported by rigorous theory. Nevertheless, taking these models together makes salient the qualitative shifts in logical understanding that students must achieve in order to understand mathematical proof. University-level mathematics is more rigorous than school mathematics. Moreover, it is precisely the formal logical structure that students must understand. All proofs, even the most ‘Cartesian’, have formal logical structure. The chequerboard proof, to return to this example, is a proof by contradiction (‘assume that the mutilated chequerboard has a complete domino tiling...’). What would be useful, at this stage, is a tool that helps students to focus on the formal logical structure of proofs. This is where an interactive theorem prover (ITP) can help (see Hanna, Larvor, & Yan, 2023; Hanna & Yan, 2021; Thoma & Ianonne, 2021; Bartzia et al., 2023). Our exploratory study investigated how well undergraduate students were responding to this novel teaching approach and how they were able to explore ways to construct a proof. Large-scale empirical studies are needed to fully understand and determine the extent to which this new approach to the teaching of proof is effective. Of course, teaching proof with the use of a proof assistant at the undergraduate level might require new and explicit instructional strategies (Balacheff & de la Tour, 2019).

### **3. Experimenting with the proof assistant Lean: An exploratory study**

The purpose of this study is to provide empirical evidence of how undergraduate students construct proofs with the Lean theorem prover, and to seek new insights into effective methods for the teaching and learning of proof that are reflective of current mathematical practice in its growing use of digital technology. To this end, we used an exploratory study method to examine how three undergraduate students approach the proof of double negation with Lean to address the following research questions:

- What are the ways undergraduate students use to prove double negation with Lean?
- To what extent does using Lean allow for autonomy, flexibility, and creativity?
- What might students gain when proving with a theorem prover?

#### **3.1 A short description of Lean**

Lean, as a theorem prover and a programming language, is used to construct and verify mathematical proofs and is freely available. In contrast to paper-and-pencil proofs, the Lean environment offers unique features for proof construction. Before we elaborate on what Lean can do and what features it has to offer, it is important first to note what Lean does not do.

Firstly, Lean does not automatically generate a proof of a particular theorem for a user. In other words, Lean does not deliver a proof of any theorem one requests. Instead, a Lean user must write commands that Lean understands so that Lean can build and check each step of the proof for the user. Secondly, since one has to build a proof using a particular method or strategy, it is often the case that Alice’s Lean proof of a particular theorem is different from Bob’s Lean proof of the same theorem. That is, the length, method, and level of elegance of a Lean proof are determined by the user. Lean’s role is to determine the correctness of each step the user builds along the way. Whether a Lean proof is short or long, once Lean indicates that “goals accomplished” or “no goals”, the proof is guaranteed to be flawless<sup>6</sup>.

A Lean proof consists of a sequence of commands, known as *tactics* to be entered on the left-hand side of the screen. The right-hand side presents a *tactic state* where Lean communicates with the user about the current goal, the status of a reasoning step, and the validity of a tactic entered. Figure 1 shows a Lean proof that  $P \rightarrow (Q \rightarrow P)$ ,<sup>7</sup> consisting of three commands in blue ovals on the left and the *Tactic state* on the right. The arrows indicate the current goal of the proof resulting from the command entered. The texts in green enclosed by `/-` and `-/` are comments or notes one wishes to add. They are meant for humans to read and are ignored by Lean.

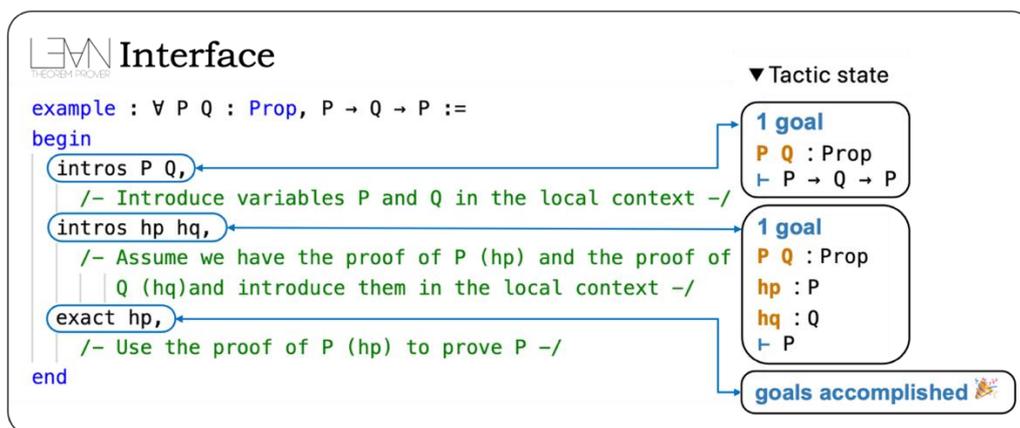


Figure 1. The Lean interface, adapted from Yan and Hanna (2023)

In addition to the dynamic feature between *tactics* and *Tactic state*, Lean also allows one to present a gapped proof with a particular proof structure using the placeholder “sorry” as shown in Figure 2. From a pedagogical perspective, a task for students to complete could be to fill in one of the “sorrys”. A gapped Lean proof is flexible in that it does not affect or

<sup>6</sup> This is due to the Lean Mathematical Library <https://leanprover-community.github.io/mathlib-overview.html>, a community-driven collective effort to build a unified library of formalized mathematics. Over 300 contributors to the library have formalized more than 73K definitions and 134K theorems across a dozen mathematics topics in undergraduate mathematics, such as linear algebra, group theory, vector space, calculus, real analysis, and complex analysis.

<sup>7</sup>  $P \rightarrow (Q \rightarrow P)$  in Lean is written as  $P \rightarrow Q \rightarrow P$  since implication is right-associative in Lean, meaning  $P$  implies that  $Q$  implies  $P$ .

disrupt Lean verifying the correctness of the proof if one chooses to work on the second “sorry” and return for the first later.

```
example (P : Prop): ¬ ¬ P → P :=
begin
  by_cases p : P,
  { sorry, },
  { sorry, },
end
```

Figure 2. A gapped proof of double negation

It is possible with the help of Lean to prove both logical statements and mathematical theorems as shown in the well-known book “How to Prove It” by Daniel Velleman, first published in 1994. It has recently been updated by adding Lean proofs to the examples in the original book. The updated, but incomplete version, is accessible online.<sup>8</sup>

A starting point for learning Lean proofs is to play games. The “Natural Number Game”<sup>9</sup> is a well-organized tutorial, focusing on Peano’s axioms and the principle of mathematical induction to build proofs of a wide range of basic facts about natural numbers, sets, and propositional logic. For example, prove that:

- $0 + n = n$
- The distributive law:  $t(a + b) = ta + tb$
- $(a * b) ^ n = a ^ n * b ^ n$
- $p \rightarrow q \rightarrow (\neg q \rightarrow \neg p)$
- If  $a \leq b, b \leq c$ , then  $a \leq c$ .

Similarly, the “Set Theory Game”<sup>10</sup> focuses on the basics of theorem proving in Lean about unions, intersections, and complements of sets. Other resources to engage in Lean proofs and teaching with Lean are well documented on the Lean Community site<sup>11</sup>. In particular, the “Lean for Teaching” stream on ZulipChat<sup>12</sup> includes reports and lively discussions on the benefits and challenges of teaching with Lean.

---

<sup>8</sup> [How to prove it with Lean](#) by Velleman

<sup>9</sup> The Natural Number Game is available in Lean 3

([https://www.ma.imperial.ac.uk/~buzzard/xena/natural\\_number\\_game/index2.html](https://www.ma.imperial.ac.uk/~buzzard/xena/natural_number_game/index2.html)) and Lean 4

(<https://adam.math.hhu.de/#/g/leanprover-community/nng4>), developed by developed by Kevin Buzzard and Mohammad Pedramfar

<sup>10</sup> The Set theory Game: <https://adam.math.hhu.de/#/g/djvelleman/stg4>, developed by Daniel Velleman, Alexander Bentkamp, Jon Eugster, and Patrick Massot

<sup>11</sup> <https://leanprover-community.github.io/>

<sup>12</sup> [Lean for Teaching](#) on ZulipChat; [Formalization and mathematics](#) by Matthew Ballard; [Proving equalities in Lean](#) by Heather Macbeth; [Introduction aux mathématiques formalisées](#) by Patrick Massot

## 3.2 Study design and procedure

### 3.2.1 Lean seminar series

To introduce Lean to undergraduate students, our research team developed teaching materials and conducted the Lean seminar series for first-year undergraduate students enrolled in *Introduction to Proofs* and *Foundation of Computer Science* courses. The *Introduction to Proofs* course focuses on abstraction, rigor, logic, and proofs by providing opportunities for students to practice reading and understanding mathematical statements, analysing definitions and properties, formulating conjectures and generalizations, providing and writing reasonable and precise arguments, and writing and critiquing proofs. The *Foundation of Computer Science* course has a logic and proof component situated in computer science, focusing on propositional and predicate logic, as well as theorems, proofs, and algorithms in number theory. The two courses explicitly aimed at introducing formal logic and proof techniques and helping students develop reasoning skills and proof competency. This makes them highly relevant to proof and proving with Lean.

The Lean seminar series, consisting of ten one-hour sessions, was presented in person over ten weeks. Participation in the seminar was optional. The first session focused on an introduction to logical operations, universal and existential quantifiers, Peano’s axioms, and the Natural Number Game designed for newcomers to Lean. Although the Natural Number Game hardly requires any prerequisite knowledge or skills, it takes from one week up to one month to complete all 69 levels of the ten sections of the game, depending on how committed a student might be.

Starting with the second session and onward, each session focused on one Lean *tactic* at a time to help students familiarize themselves with a specific tactic and in what mathematical contexts the tactic can be helpful. As mentioned in section 3.1, *tactics* are commands that give Lean information on the path to follow in building proofs. Lean performs *tactics* (that the user entered) and subsequently modifies a proof goal state. A Lean proof, therefore, consists of a combination of a sequence of *tactics*.

Commonly-used *tactics* in Lean:

- The “intro” *tactic* introduces a universally quantified variable or a proof of a hypothesis in the local context.
- The “apply” *tactic* takes a lemma or theorem, matches the conclusion with the current goal, and leaves the hypotheses, if any, as new goals.
- The “by contra” *tactic* negates what needs to be proven by bringing an additional hypothesis  $h : \neg P$ . This leads to the change of the goal, from proving  $P$  to proving false, a contradiction.
- The “by\_cases” *tactic* splits the main goal into two cases, assuming  $h : P$  in the first branch, and  $h : \neg P$  in the second branch.

- The “induction  $n$  with  $d$   $hd$ ” *tactic* attempts to prove the goal by induction on  $n$ , with the inductive assumption in the successor case being  $hd$ . Specifically, it turns the goal into two goals, a base case with  $n = 0$  and an inductive step where  $hd$  is a proof of the  $n = d$  case and the goal is the  $n = \text{succ}(d)$  case (see an example in the Appendix).

Each session also provided students with examples and exercises that allowed them to work in groups or individually. As the seminar series progressed, the existing attendees brought other students interested in Lean but enrolled in other courses. Over the ten weeks, students who majored in mathematics, computer science, life sciences, and chemistry attended the seminar series.

### 3.2.2 Participants and task

After the Lean workshop, three undergraduate students majoring in mathematics with a computer science minor participated in task-based interviews. All three participants had experience in programming and some fluency in programming languages such as Python. Though the syntax of the Lean theorem prover differs from the programming languages they were familiar with, their experiences in programming contributed to a reasonably easy transition to Lean. Within the twelve weeks from the first session of the Lean workshop to the time that interviews were conducted, the participants have learned more than a dozen Lean *tactics* and constructed 30 Lean proofs on average. This paper focuses on analysing and discussing proofs of double negation created by the three participants, Amir, Adam, and Ava.

Proof of double negation was covered in the *Introduction to Proofs* and *Foundation of Computer Science* courses. Students learned how to prove it in a paper-and-pencil format. Prior to the interviews, the participants had an opportunity to review what they had learned from the Lean workshop, and they were then asked to use Lean to prove double negation ( $\neg\neg P \rightarrow P$ ), while thinking aloud. We observed each of the participants while they were using Lean to construct their assigned proof. After they finished, we asked the participants to reflect on the approaches they used and the decisions they made and conducted individual interviews to elicit their thoughts. We were looking for three main things during the interviews: what was going through their minds as they were writing a Lean proof, how they would explain a step of their proof written in Lean code, and what they think about their proof looking back. All interviews were audio-recorded and transcribed.

## 4. Findings

Figure 3 shows five different ways of proving double negation presented by the three students. Although the students were not asked to prove double negation in multiple ways, Adam and Ava gave two different proofs, and Amir offered one. It is worth noting that proof 5 is a term proof, which is a different type of proof one can produce in Lean. It still uses *tactics* but does not require a ‘begin...end’ block. Without a ‘begin...end’ block, Lean considers a term-proof as one argument and only evaluates when one reaches the end. This makes Lean’s feedback less helpful during the proving process. In what follows, we focus on proofs 1, 2, and 4, one from each student, to show the different approaches they took.

example (P : Prop) : ¬ ¬ P → P :=				
'by_contra' proofs		'by_cases' proofs		A term proof
Proof 1	Proof 2	Proof 3	Proof 4	Proof 5
<pre>begin   change ((P → false) → false) → P,   intro h,   by_contra,   exact h1 h, end</pre>	<pre>begin   intro dnp,   by_contra,   apply dnp,   exact h, end</pre>	<pre>begin   intro nnp,   by_cases p,   exact h,   exfalso,   exact nnp h, end</pre>	<pre>begin   by_cases p : P,   {intro dnp,    exact p,},   {intro dnp,    exfalso,    contradiction,}, end</pre>	<pre>by_contradiction   (assume h1 : ¬ P,    show false, from h h1)</pre>
Adam	Amir	Adam	Ava	Ava

Figure 3. An overview of five student-generated proofs

### Amir's proof

The method Amir used to prove double negation was to use proof by contradiction. Figure 4(1) shows a clear path of this approach. He first introduced the hypothesis of double negation that  $\neg\neg P$  is true, named 'dnp'. He then used the 'by contra' tactic, which stands for the method of proof by contradiction. The tactic essentially tells Lean to assume that the negation of the current goal ( $\neg P$ ) is true and to prove it false, equivalent to finding a contradiction. As Amir reached this step, he was unsure how to proceed. He could have proceeded by observing that the two hypotheses, 'dnp :  $\neg\neg P$ ' and 'h :  $\neg P$ ', were, in fact, contradictory, and then used the 'contradiction' tactic to tell Lean that a contradiction is found to complete the proof.

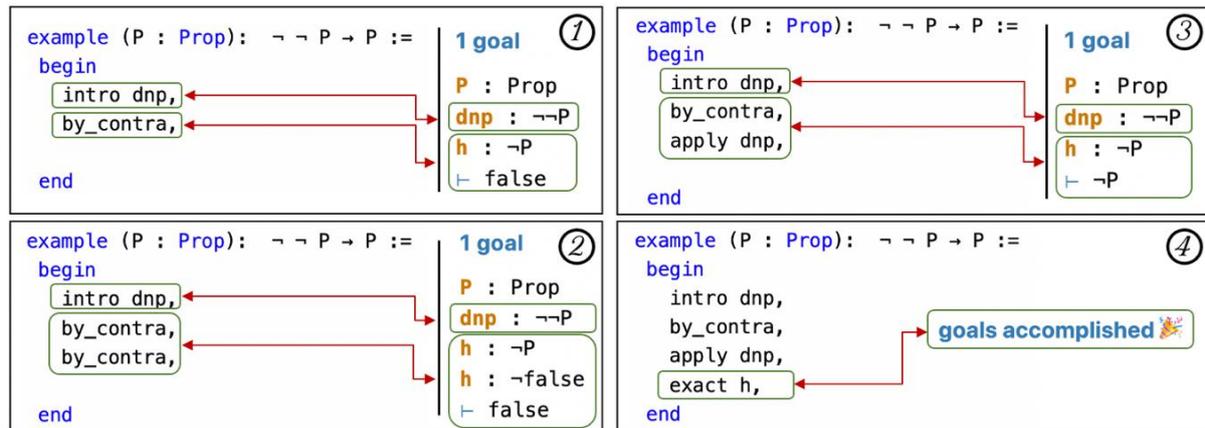


Figure 4. Amir's proof of double negation

Instead, Amir tried out the 'by\_contra' tactic again to see what might happen (see Figure 4(2)). Lean now assumes the negation of the current goal ( $\neg false$ ) is true and prove 'false', a contradiction. "Umm, OK. So that doesn't look great," Amir commented on his experiment. Although the new hypothesis 'h :  $\neg false$ ' seems inconvenient to work with, the attempt indeed works for the same reason mentioned above regardless of the repeated use of 'by contra'. Nevertheless, Amir went in a different direction, as shown in Figure 4(3): he replaced the second 'by\_contra' with 'apply dnp'.

Using the apply tactic requires an understanding of how negation is defined in Lean. Amir explained:

So my goal was to prove false. My hypothesis ‘dnp’, which is not not P, is defined as not P implies false. Since I have not P implies false and my goal is false, then I can shift the goal by using that implication, shifting it from false to not P. Since I have that implication and it [false] is sort of hidden under ‘dnp’...

Amir intended to unpack ‘dnp :  $\neg\neg P$ ’ using the definition. Namely, ‘dnp :  $\neg\neg P$ ’ becomes ‘dnp :  $\neg P \rightarrow \text{false}$ ’. Although this exchange only occurred in Amir’s mind, it allowed him to make a connection between the implication (dnp :  $\neg P \rightarrow \text{false}$ ) and the current goal (false) by using the apply tactic. That is, ‘apply dnp’ matches the conclusion of the implication (false) with the current goal (false) and leaves the hypothesis ( $\neg P$ ) as the new goal. To prove  $\neg P$  is true, it is rather straightforward due to ‘h :  $\neg P$ ’, which reads: h is the proof or hypothesis of  $\neg P$ . Figure 4(4) shows that exact h accomplished the goal.

### Adam’s proof

Adam’s first attempt to prove double negation was to use the definition of negation in Lean. He looked at the statement of double negation on his screen and started to talk about negation in Lean:

So, I have learned a bit about the ‘not’. I know the definition of ‘not P’ is P implies false. So this [ $\neg\neg P$ ] would be not P implies false [ $\neg P \rightarrow \text{false}$ ], which the whole thing would be P implies false implies false [( $P \rightarrow \text{false}$ )  $\rightarrow$  false], I think. But that’s kind of hard to think about. Since it’s an implication, I’d just start with the intro thing. (See Figure 5(1))

<pre>example (P : Prop): <math>\neg\neg P \rightarrow P :=</math> begin   intro hnp, end</pre>	<pre>1 goal P : Prop hnp : <math>\neg\neg P</math> ├ P</pre>	①
<pre>example (P : Prop): <math>\neg\neg P \rightarrow P :=</math> begin   change P <math>\rightarrow</math> false, end</pre>	<pre>tactic.change failed, given type P <math>\rightarrow</math> false is not definitionally equal to <math>\neg\neg P \rightarrow P</math> state: P : Prop ├ <math>\neg\neg P \rightarrow P</math></pre>	②
<pre>example (P : Prop): <math>\neg\neg P \rightarrow P :=</math> begin   change ((P <math>\rightarrow</math> false) <math>\rightarrow</math> false) <math>\rightarrow</math> P, end</pre>	<pre>1 goal P : Prop ├ ((P <math>\rightarrow</math> false) <math>\rightarrow</math> false) <math>\rightarrow</math> P</pre>	③
<pre>example (P : Prop): <math>\neg\neg P \rightarrow P :=</math> begin   change ((P <math>\rightarrow</math> false) <math>\rightarrow</math> false) <math>\rightarrow</math> P,   intro h,   by_contra h1, end</pre>	<pre>1 goal P : Prop h : (P <math>\rightarrow</math> false) <math>\rightarrow</math> false h1 : <math>\neg P</math> ├ false</pre>	④

Figure 5. Adam’s partial proof of double negation

After Adam introduced the hypothesis that  $\neg\neg P$  is true, called ‘hnp’, he was stuck:

OK, so I want to prove  $P$  from not not  $P$ . So I think I want to rewrite what this  $[\neg\neg P]$  is because it's kind of hard to look at. Which tactic is it for this? I am trying to remember. Oh, the change tactic. It works on the goal. How about I use change before I do anything? OK, so I want change  $P$ . Let's see. (See Figure 5(2))

Adam's attempt to use 'change  $P \rightarrow \text{false}$ ' led to a message from Lean that indicated a failure of the tactic since ' $P \rightarrow \text{false}$ ' is not definitionally equal to ' $\neg\neg P \rightarrow P$ '. "Maybe I need to, like fully expand it," he said to himself. The second attempt successfully changed the goal as shown in Figure 5(3). Adam then took the same path as Amir in Figure 4(1), using the 'intro' and 'by contra' tactics shown in Figure 5(4). What made Adam's proof different at this stage was that the hypothesis  $h$  could be considered as a function that maps from ' $P \rightarrow \text{false}$ ' to 'false'. This makes proving false handy, as Adam explained:

We have the proof of not  $P$  [ $h1 : \neg P$ ] and we want to show that we get a contradiction. And  $\neg P$  is equal to ' $P \rightarrow \text{false}$ '. So if I evaluate this function [ $h : (P \rightarrow \text{false}) \rightarrow \text{false}$ ], I'll just get what I need immediately.

To show false, Adam used 'exact  $h1\ h$ ', which evaluates  $h1$  at  $h$ , to close the goal.

### Ava's proof

One way to prove  $\neg\neg P \rightarrow P$  in a paper-and-pencil format is to use a truth table to map out the truth values of  $\neg\neg P$ . Ava applied this method for her Lean proof of double negation:

I think I know the tactic is 'by\_cases'. I'd like to split  $P$  into two cases,  $P$  is true and  $P$  is false. It is essentially using truth table. If I do 'by\_cases  $p$ ', oh gosh, did it not work? Maybe I have the syntax wrong. Ok, I guess I need to identify. (See Figure 6(1))

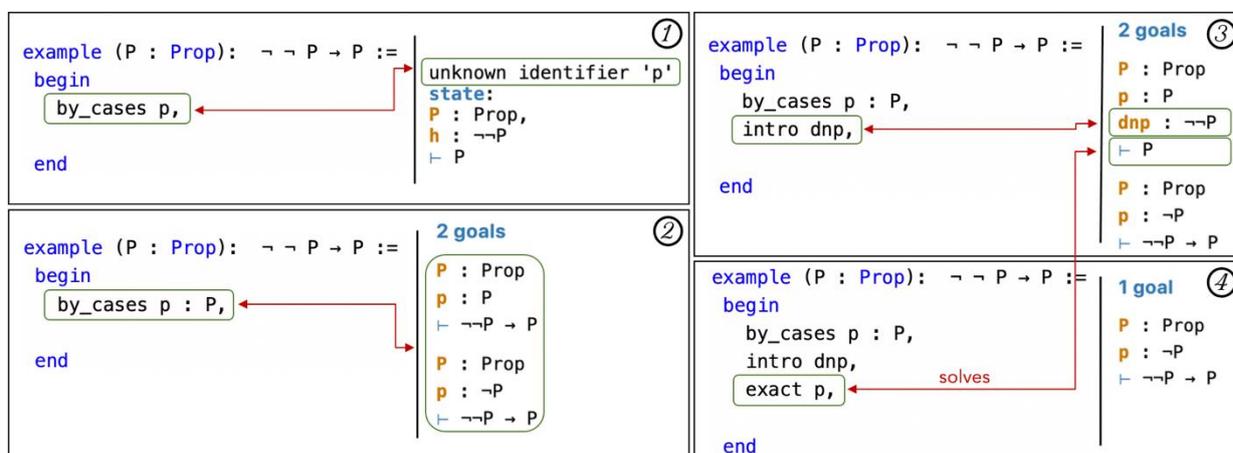


Figure 6. Ava's partial proof of double negation

Once Ava specified the name of the new hypothesis  $p$  using the syntax 'by\_cases  $p : P$ ', Lean assumes ' $p : P$ ' in the first branch and ' $p : \neg P$ ' in the second branch. Subsequently, Lean creates two goals based on the two cases (See Figure 6(2)). To solve the first goal, after introducing the hypothesis 'dnp' that  $\neg\neg P$  is true shown in Figure 6(3), Ava used 'exact  $p$ ' since ' $p : P$ ' was given, shown in Figure 6(4). There are multiple ways to solve the second goal, using 'by\_contra' or 'contradiction', for example, in Amir and Adam's proofs. Ava chose to use the 'exfalso' tactic that turns the target of the current goal to false so that she

could claim a contradiction – an alternative strategy to approach proofs involving negation. She explained, “If I apply not P to not not P, I would get a contradiction. So I first need to use ‘exfalso’ to turn my goal to false and then say contradiction.” The structure of Ava’s proof is shown in Figure 7.

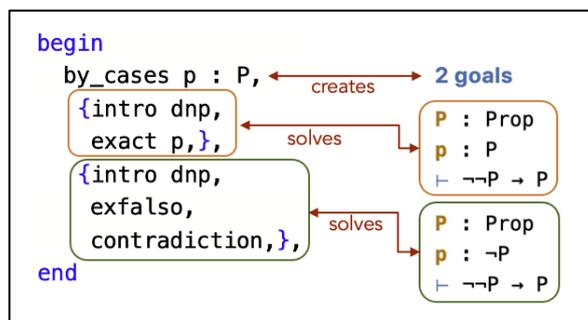


Figure 7. The structure of Ava’s Lean proof

## 5. Discussion

The process of mathematical reasoning involves applying logical rules and operations to manipulate and analyze mathematical statements, allowing students to establish relationships between statements, draw conclusions, and derive new theorems from existing ones. Students are typically taught various techniques, such as direct proofs, proof by contradiction, mathematical induction, and proof by cases, that help them construct rigorous and valid arguments. As stated above by Tall (2008), Harel and Sowder (1998), Harel (2007), Durand-Guerrier et al. (2012), and Tao (2009), and revealed in many more empirical studies, undergraduates can and do make the qualitative shift to an emphasis on a formal framework of proving. Nevertheless, understanding the formal structure of mathematical proofs is a significant cognitive advance, and any assistance must be welcome. In this paper, we discussed compelling conceptual and cognitive grounds for the addition of a new technique: the use of an interactive theorem prover, Lean, based on formalized mathematics. The findings of our exploratory study, admittedly tentative, suggest that the rigorous nature of Lean is not an obstacle for students and does not seem to lead to stifling students’ creativity in proving. On the contrary, proving with Lean offers a great deal of flexibility - there are multiple paths that one can take to reach the same conclusion.

As the findings show, the three students approached the proof of double negation in different ways. This suggests that Lean allowed for autonomy, flexibility, and creativity as the students used proof strategies that immediately came to their minds and chose the tactics that helped them proceed in that particular direction thus moving to “the formalism of advanced mathematical thinking” (Tall, 2008) and using the rules of logic (Durand-Guerrier et al., 2012). The free choices of paths and tactics allow them to experiment and treat a proof as a puzzle or maze. Perhaps Amir might not have tried nested sub-proofs by contradiction if he had been working with pencil and paper. Similarly, Ava seems to have been prompted to consider a proof by cases simply because she had ‘by cases’ available as a Lean tactic. Mathematics is always inscribed one way or another (as diagrams drawn in wet sand, in chalk on a board, or in Latex code in an email, for example). The medium of inscription recommends some kinds of argument and inhibits others. For example, a whiteboard facilitates small diagrams and so encourages spatial reasoning. Writing in a digital text-editor

(such as email) precludes diagrams but makes copying and pasting chunks of text very easy, which encourages exploiting modularity. Writing arguments in Lean, with a library of tactics available, invites students to experiment with proof ideas that they might not have considered otherwise.

We claim that what is gained in proving with Lean is the adventurous aspect of proving and the autonomy to go about a proof in one's own way. Of course, one can write whatever one likes with a pencil and paper. However, the slow process of having one's work checked by a human expert makes it difficult to experiment freely. As Lean provides instant feedback with no shame attached, it allows students to play with proof ideas. This playfulness would likely enhance student engagement with proof, and further change how they perceive the creation and presentation of proof. It would also lead students to feel "comfortable with the rigorous foundations" (Tao, 2009) of Lean and would also give support to Kevin Buzzard's observation that:

Arguments in lectures may take place high above the axioms, with technical details being dismissed as obvious or easy to verify, and left to the reader (perhaps with some hints). This is the beginning of what Terry Tao [Tao09] has called the post-rigorous stage of mathematics. (Buzzard, 2020, p. 2)

At this stage, students note that when they try to translate a mathematical argument to a format suited for automated processing, they confront a mismatch between a text-book presentation and what is required by the interactive theorem prover (ITP). This is indeed a step up and cognitive leap from Tao's (2009) pre-rigorous stage to rigorous stage. Ordinary school mathematics is simply too informal for the logical and algebraic procedures that ITPs demand. Subtle contextual cues are needed to resolve the ambiguities that are part of informal mathematics, requiring not just knowledge of the notation and conventions but a thorough understanding of the relevant mathematical content. The fact that students were able to explain their reasoning to both Lean and the human interviewer, indicates that they had a good understanding of the proof they were dealing with. There are many instances in mathematics where we thought we understood a concept or method quite well until we talked to someone else or did it in a different way, we then realized that we now understand it better. It is not always about learning something new but rather understanding something at a deeper level. This is the benefit that all mathematics educators care about, and we believe that Lean has the potential to offer it.

We insist that informal mathematical reasoning is an essential aspect of mathematical exploration and discovery and has been shown to constitute an important element of mathematics education. The introduction of formalization to the teaching of mathematics at the undergraduate level is not meant to replace informal mathematical reasoning, but rather to complement it. As we mentioned in sections 1 and 2, all proofs, even the most 'Cartesian', have a formal logical structure. Further research might show that the activity of creating proofs in Lean preserves or perhaps even deepens some of the advantages associated with 'Cartesian' proofs. In particular, we concede that our limited exploratory study doesn't allow us to determine what role Lean (or any other theorem prover) can play in helping to construct proofs that are explanatory, that is, proofs that show not only "that" a statement is true but also "why" it is true. To bring out the explanatory value of a proof, it might be necessary to write it in a more 'Cartesian' style, that is, suppressing most of the logical machinery so that the central explanatory idea is easier to appreciate. It's worth noting, though, that this style of writing, as exemplified by Aigner & Ziegler (2010), is suitable for readers who can supply

the suppressed logical structure without having to think about it. In other words, it is suitable for readers who have reached and perhaps passed through Tao's second stage. It's worth noting that the use of Lean involves frequent movements between the language of Lean and mathematics using pencil and paper. There is, therefore, no danger of students losing sight of the mathematical meaning of what they're doing. Or if they ever do, they soon get stuck and can only get going again by thinking about the meaning of the symbols in their formalization (as happened several times to the students in our study).

The students in this study were all self-selecting and all used to coding. This is often the case in the few existing studies of teaching with Lean, for example, Thoma & Iannone (2021). As trends indicate that interactive theorem proving will eventually make it to mainstream mathematics, it will become necessary to introduce proof assistants to undergraduate mathematics as soon as possible and to open a discussion on the ways they would be best used. Our exploratory study showed some evidence that students do learn how to use a proof assistant, but it is not clear to what extent the proof assistant can be used as a teaching tool. Large-scale empirical research on the learning of mathematical proof with a proof assistant will no doubt contribute to our understanding of the potential benefits and challenges that the new technology presents.

## Acknowledgement

We wish to acknowledge the generous support of the Social Sciences and Humanities Research Council of Canada. We also thank the anonymous reviewers for their valuable comments and suggestions.

## References

- Aigner, M., & Ziegler, G. M. (2010). *Proofs from The BOOK* (4th ed.). Springer.
- Alekhnovich, M. (2004). Mutilated chessboard problem is exponentially hard for resolution. *Theoretical Computer Science*, 310(1-3), 513-525.
- Avigad, J. (2023). Mathematics and the formal turn. *Bulletin (New Series) of the American Mathematical Society*. <https://arxiv.org/pdf/2311.00007.pdf>
- Balacheff, N., & de la Tour, T. B. (2019). Proof technology and learning in mathematics: Common issues and perspectives. In G. Hanna, D. Reid, & M. de Villiers (Eds.), *Proof Technology in Mathematics Research and Teaching* (pp. 349-365). Springer.
- Bartzia, E. I., Beffara, E., Meyer, A., & Narboux, J. (2023). Proof assistants for undergraduate mathematics education: Elements of an a priori analysis. <https://hal.science/hal-04087080v1/document>
- Buzzard, K. (2020). Proving theorems with computers. *Notices of the American Mathematical Society*, 67(11), 1791-1799. <https://www.ma.imperial.ac.uk/~buzzard/xena/computers2.pdf>
- Castelvecchi, D. (2021). Mathematicians welcome computer-assisted proof in “grand unification” theory. *Nature*, 595, 18–19.
- Durand-Guerrier, V. (2003). Which notion of implication is the right one? From logical considerations to a didactic perspective. *Educational Studies in Mathematics*, 53, 5–34. <https://doi.org/10.1023/A:1024661004375>

- Durand-Guerrier, V., Boero, P., Douek, N., Epp, S.S. & Tanguay, D. (2012). Examining the role of logic in teaching proof. In G. Hanna & de Villiers (Eds.), *Proof and proving in mathematics education*, 369—389. Springer.
- Epp, S. S. (2003). The role of logic in teaching proof. *The American Mathematical Monthly*, 110(10), 886-899.
- Gowers, T., Barrow-Green, J., & Leader, I. (Eds.). (2010). *The Princeton companion to mathematics*. Princeton University Press.
- Granville, A. (Forthcoming). Proofs: Objective truth vs. culturally robust. *Annals of Mathematics and Philosophy*.
- Grenier, D., & Payan, C. (2006, May). Des «situations recherche» pour l'apprentissage des savoirs transversaux. In *L'enseignement des mathématiques face aux défis de l'école et des communautés. Les actes du colloque EMF*.
- Hacking, I. (2014). *Why is there philosophy of mathematics at all?* Cambridge University Press.
- Hales, T. C. (2012). *Dense sphere packings: A blueprint for formal proofs* (Vol. 400). Cambridge University Press.
- Hamkins, Joel D. (2020) *Lectures on the philosophy of mathematics*. MIT Press.
- Hanna, G., & Yan, X. (2021). Opening a discussion on teaching proof with automated theorem provers. *For the Learning of Mathematics*, 41(3), 42–46.
- Hanna, G., Larvor, B., & Yan, X. (2023). Human-machine collaboration in the teaching of proof. *Journal of Humanistic Mathematics*, 13(1).  
<https://scholarship.claremont.edu/cgi/viewcontent.cgi?article=1980&context=jhm>
- Hardy, G. H. (1929). Mathematical proof. *Mind, New Series*, 38(149), 1-25.
- Harel, G., & Sowder, L. (1998). Students' proof schemes, In E. Dubinsky, A. Schoenfeld & J. Kaput (Eds.), *Research on collegiate mathematics education*, AMS, vol. III, (pp. 234-283).
- Harel, G. (2007) Students' proof schemes revisited. In P. Boero (Ed), *Theorems in schools: From history, epistemology and cognition to classroom practice* (pp. 65-78). Sense.
- Littlewood, J. E. (1953). *A mathematician's miscellany*. London: Methuen.
- Pinto, M. M. F. (1998). *Students' understanding of real analysis*. Unpublished PhD, Warwick University.
- Tall, D. (2008). The transition to formal thinking in mathematics. *Mathematics Education Research Journal*, 20, 5–24 <https://doi.org/10.1007/BF03217474>
- Tanswell, F. (2015). A problem with the dependence of informal proofs on formal proofs. *Philosophia Mathematica*, 23(3), 295-310.
- Tao, T. (2009). There's more to mathematics than rigour and proofs. Blog.  
<https://terrytao.wordpress.com/career-advice/theres-more-to-mathematics-than-rigour-and-proofs/>

- 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*, 8, 64–93. <https://doi.org/10.1007/s40753-021-00140-1>
- Voevodsky, V. (2014). The origins and motivations of univalent foundations. *IAS, The Institute Letter*, 8–9. <https://www.ias.edu/sites/default/files/documents/publications/ILsummer14.pdf>
- Weber, K. (2004). Traditional instruction in advanced mathematics courses: A case study of one professor’s lectures and proofs in an introductory real analysis course. *Journal of Mathematical Behavior*, 23, 115–133.
- Wittgenstein, L. (1956). *Remarks on the foundations of mathematics*. Tr. G.E.M. Anscombe. Oxford: Blackwell.
- Yan, X., & Hanna, G. (2023). Using the Lean interactive theorem prover in undergraduate mathematics, *International Journal of Mathematical Education in Science and Technology*. DOI: 10.1080/0020739X.2023.2227191

**Appendix:** Tactics used and changes in goals in a proof of  $0+n=n$

To prove that if  $n$  is a natural number, then  $0+n=n$ .

`example (n : ℕ) : 0 + n = n :=`

Tactic Used	Note	Tactic State
<code>begin</code>		<b>1 goal</b> <code>n : ℕ</code> <code>⊢ 0 + n = n</code>
<code>induction n with n ih,</code>	Turn the current goal into 2 goals: a base case <code>nat.zero</code> and an inductive case <code>nat.succ</code> .	<b>2 goals</b> <code>case nat.zero</code> <code>⊢ 0 + 0 = 0</code> <code>case nat.succ</code> <code>n : ℕ</code> <code>ih : 0 + n = n</code> <code>⊢ 0 + n.succ = n.succ</code>
<code>norm_num,</code>	The <code>norm_num</code> tactic normalizes numerical expressions such as $0 + 0 = 0$ , and therefore closes the first goal.	<b>1 goal</b> <code>case nat.succ</code> <code>n : ℕ</code> <code>ih : 0 + n = n</code> <code>⊢ 0 + n.succ = n.succ</code>
<code>rw add_succ,</code>	' <code>add_succ</code> ' refers to $\forall (n m : \mathbb{N}), n + m.succ = (n + m).succ$ . The <code>rw</code> tactic replaces <code>0 + n.succ</code> with <code>(0 + n).succ</code> .	<b>1 goal</b> <code>case nat.succ</code> <code>n : ℕ</code> <code>ih : 0 + n = n</code> <code>⊢ (0 + n).succ = n.succ</code>
<code>rw ih,</code>	Replace <code>0 + n</code> with <code>n</code> to obtain <code>n.succ = n.succ</code> .	<b>goals accomplished</b> 🎉
<code>end</code>		

- The “induction” *tactic* proves the goal by induction on  $n$ , with the inductive assumption in the succ case (successor case) being “`ih`”. Specifically, “`induction n with d hd`” turns the goal into two goals, a base case with  $n = 0$  and an inductive step where “`hd`” is a proof of the  $n = d$  case.

- The “norm\_num” *tactic* normalized any numerical expressions.
- The “rw” (rewrite) *tactic*: given a hypothesis (h) of the form  $A = B$ , “rw h” replaces occurrences of A with B.