

A Data-Driven Strategy for Integrating Lean 4 into the EPFL Undergraduate Mathematics Curriculum

Jérémy Digard

Under the supervision of **Prof. Maryna Viazovska**
Summer Internship Report

Chair of Number Theory (SB - MATH)
École Polytechnique Fédérale de Lausanne (EPFL)

August 29, 2026

Abstract

This report explores the feasibility of integrating the Lean 4 proof assistant into the Undergraduate mathematics curriculum at EPFL. Based on a data-driven self-study conducted during a summer internship under the supervision of Prof. Maryna Viazovska, this document identifies the primary roadblocks faced by beginners. Quantitative analysis reveals that over 55% of early friction points stem from syntax and state management. Consequently, while Lean acts as an exceptional tutor for enforcing logical rigor, its steep learning curve risks subjecting students to a “double cognitive load”. To prevent this, the report proposes a seamless integration strategy, coupled with visual tools like Paperproof to preserve global mathematical intuition, and a community-driven acquisition strategy (introductory workshops and “Territory Expansion” hackathons) to build momentum and test the infrastructure.

Introduction

The transition from high school to university-level mathematics is a notoriously challenging milestone for Undergraduate students. At EPFL, this shift is characterized by the sudden demand for absolute logical rigor, a standard that is difficult to calibrate using traditional pen-and-paper proofs. In recent years, interactive theorem provers, particularly Lean 4, have emerged as promising educational tools to bridge this gap. In Lean, the code itself is the mathematical proof, and the compiler’s sole job is to mechanically verify that every single deduction aligns with axioms. By providing instant, incorruptible feedback, Lean forces students to confront their logical leaps and solidifies their understanding of formal proofs.

However, if not implemented carefully, there is a risk of replacing a mathematical struggle with a software engineering one, thereby subjecting students to a double cognitive load.

Over my first month, I documented every roadblock I encountered while learning Lean, creating a dataset to measure the friction points of the software.

This report begins with a quantitative analysis of my own learning curve. Building on these insights, it then explores the broader pedagogical value of Lean, contrasting the traditional “illusion of rigor” found in paper-based proofs with the cognitive scaffolding provided by the software. Finally, the report outlines a practical implementation strategy focused on minimizing technical friction and fostering a strong student community around the tool.

1 Quantitative Analysis of Beginner Roadblocks in Lean

During my first month using Lean, I archived all my roadblocks in my exchanges with my AI assistant (the AI acted purely as an on-demand Lean manual for technical hurdles). To draw conclusions from this, I developed a script to automatically analyze and categorize this data. The goal was to identify trends and the various frictions I faced [8].

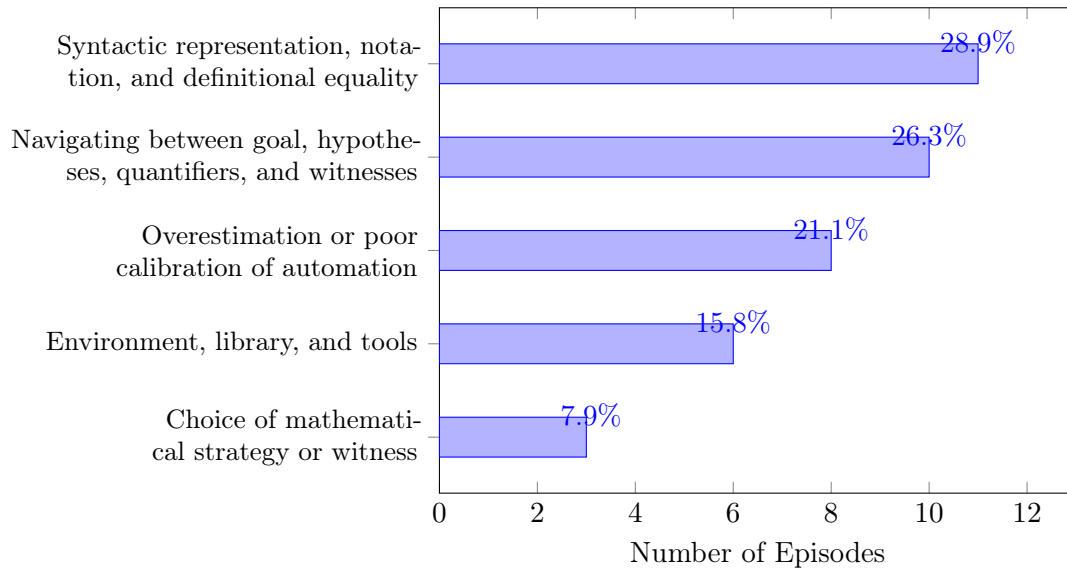


Figure 1: Distribution of Beginner Roadblocks Encountered During the First Month

1.1 The Core of the Problem: Building a Mental Model

My data shows that more than half of the episodes (55.2% - the first two rows) relate to how Lean exactly represents a proposition, and how a tactic transforms the current state. The challenge is to build a mental model of the tool. Only 7.9% of my roadblocks concerned the choice of mathematical strategy.

A. The Friction Between Mathematical Equivalence and Syntactic Strictness. As a student, I tended to assume that Lean should automatically recognize the mathematical equivalence behind the syntax. For example, my mathematical intuition considers that $-1 / x$ and $-(1 / x)$ are identical, just as it considers a natural number and its real counterpart to be the same, often ignoring the need for explicit coercions from $\mathbb{N}$ to $\mathbb{R}$. However, for a Lean tactic, these are syntactically different expressions. I often confused definitional equality (what is true by definition for Lean) and proven equivalence (which requires invoking a theorem to rewrite the expression).

B. Mastering the Flow of Reasoning: Forward vs. Backward Proofs. A large partition of my initial roadblocks concerned the direction of reasoning. On paper, a first-year student often writes fluidly from top to bottom. In Lean, one must precisely understand the flow of information:

- Distinguishing between **apply** (which operates backward reasoning on the goal) and **rw** (which transforms an existing expression, with a left/right direction).
- Understanding how to extract a witness with **use** or destruct an implication with **cases**.
- Distinguishing between **using** an existence already known in the hypotheses (e.g., extracting an N with **obtain**) and **proving** an existence required by the goal (providing the right N with **use**).

C. Demystifying “Magic Tactics”: The Necessity of Explicit Proof Steps. I expected a tactic to accomplish the entire chain of reasoning. My data shows that I often expected `field_simp` to normalize fractions, or `bound` to deduce on its own that a denominator is non-zero. On paper, I tend to underestimate the importance of side conditions (like proving that $\epsilon > 0$ or that $x \neq 0$ before dividing). Lean, on the other hand, forces you to prepare the ground before calling an automation tactic.

1.2 Anticipating the Double Cognitive Load for Beginners

During my learning process, I observed three distinct situations:

- On the *Natural Number Game*, the mathematical theory was easy to me. The challenge was understanding the tactics.
- On the *Real Analysis Game*, the theory was already familiar to me following my first year of undergrad. The challenge was adapting to the extreme level of detail required by Lean.
- On the *Topology Game* (which I explored on my own initiative following a discussion with Prof. Kathryn Hess, my Topology I professor for next semester, who created this game for her course), the situation was reversed: the mechanics of Lean had become familiar to me, but the mathematical theory was completely unknown. As a result, the learning process was extremely smooth (game finished in 9 hours). Lean was no longer an obstacle, but an interactive “tutor” that forced me to understand the new definitions.

Future first-year students will find themselves in a more complex situation: they will have to assimilate the new concepts of real analysis while learning Lean’s syntax. My learning experience therefore represents an ideal scenario (where either the theory is already mastered, or Lean is).

2 The Transition to University Math: The Illusion of Rigor and Lean as a Tutor

Since this project is primarily aimed at beginner students, it seems relevant to start from my own experience and that of my peers. The transition from high school to university was a major

shift for us, marked by the discovery of mathematical rigor. While this discovery is intellectually highly stimulating, I realized that learning this rigor is a constant struggle to find the right level of detail and exactness.

2.1 The Core Issue: Calibrating Rigor on Paper

When faced with a proof on paper, I often encountered the problem of the passive medium: the paper does not return any errors. In discussions with other students, I noticed that we often adopted two extreme behaviors:

- **The illusion of rigor:** I sometimes made logical leaps without realizing it, or forgot side conditions (such as dividing by a variable without checking that it is non-zero, an error very present in my own learning data). Since paper accepts anything, I was convinced my proof was valid.
- **Over-explaining:** Conversely, not knowing exactly where to set the bar for rigor, I sometimes provided unnecessary information. Out of fear of losing points, I over-detailed trivialities, which drowned out the logical structure of my proof.

Added to this is the reality of our working ecosystem at EPFL. Given the demanding workload, the myth of the student who prepares their problem sets in advance does not survive for long: in reality, during the semester, no one is ever truly ahead. Furthermore, the gap between the lecture and the exercise session is often very short, sometimes occurring on the same day. Consequently, when we arrive at the exercise session, we are still digesting the theory. We therefore use the presence of the TAs primarily to unblock our understanding of new definitions or to tackle the first direct application exercises.

The in-depth work is done in the evening, alone at home or at the library. It is precisely when we tackle the most difficult exercises that we no longer have anyone to validate our logic. Of course, the *Ed* platform allows us to ask the TAs questions, but it is an asynchronous tool. By the time I get an answer, my mental context of the problem has disappeared, and the learning loop is broken. We therefore often move forward without a safety net, risking getting stuck for a long time on false leads or, worse, assimilating and consolidating our own reasoning errors.

2.2 Lean as a Solution: The Strict Judge and the Cognitive Scaffold

Discovering Lean this summer, I realized that it offered exactly what I had been missing during my first year. What fascinated me about this tool is that it offers a different way of approaching mathematics.

A. The Rigor Tutor (The Judge). Lean acted as an incorruptible grader for me, available 24/7. It validates or invalidates every logical step instantly. This immediate feedback destroyed my “illusion of rigor”: if I made a logical leap, the compiler blocked it. I could no longer practice making mistakes. Conversely, Lean forced me to clean up my reasoning by using only what was strictly necessary to close the goal, creating a game-like reward loop that makes the process surprisingly fun.

B. The “Training Wheels” (The Cognitive Scaffold). On paper, I had to keep in mind (or search through my drafts for) all the active hypotheses and what I was trying to prove at any given moment, which created a heavy mental overload. The *Infview* solved this problem by constantly displaying the exact state of my proof, allowing me to focus on pure logic. Just as training wheels keep a bicycle balanced so the child can focus on pedaling, Lean’s interface kept my mathematical context balanced.

2.3 Fostering Mathematical Understanding: From Intuition to Formal Proof

Faced with this tool, a legitimate question arose during my learning process: does Lean actually help one *understand* mathematical theory, or does it merely verify that the lecture material has been applied?

As mathematician David Bessis writes in *Mathematica*, formal logic is not used to create intuition; it is used to clarify and validate it. Lean did not teach me the theory, but it crystallized my understanding by preventing me from settling for an illusion of obviousness. Lean does not replace the intuition conveyed by the professor, but it transforms a vague idea into a solid understanding.

A joint study conducted at Imperial [2] illustrates how this strict validation translates into student results. During a proof task on abundant numbers, the final scores speak for themselves:

Table 1: Student Performance on Abundant Numbers Proof Task [2]

Scores	Non-Lean Users	Lean Users
4	4 (13.8 %)	3 (43.0 %)
3	3 (10.3 %)	3 (43.0 %)
2	5 (17.3 %)	0 (0.0 %)
1	16 (55.2 %)	1 (14.0 %)
0	1 (3.4 %)	0 (0.0 %)

The difference is striking: 43% of Lean users achieved the maximum score, compared to only 13.8% for non-users. Lean significantly improves our ability to structure a proof.

3 Lowering the Barriers to Entry

If Lean is an excellent tutor for rigor (Section 2), how can we practically integrate it into the Undergraduate curriculum without reproducing the severe cognitive friction associated with learning a programming language (Section 1)?

3.1 Making the Tool Invisible: Zero Technical and Cognitive Friction

For Lean to fulfill its role as a tutor, a major pitfall must be avoided: too much computer science kills the mathematics. As Weber points out [3], an excessive focus on syntactic form can prevent students from understanding the deep semantic meaning of what the professor is trying to convey. If the student spends their time fighting the language architecture or the idiosyncrasies of type theory, the tool becomes a burden. Lean must act as a “logical calculator”, and not a CS course. This fading of the tool must occur on two levels: hardware and cognitive.

A. The Hardware Barrier: The Necessity of a Web Interface. The first friction point is installation. I personally lost two days trying to install Lean on my Mac, dealing with recurring bugs with the complex dependency management. Demanding this of Undergraduate students, many of whom work on tablets, creates an unnecessary technical barrier.

A possible solution is to enforce a Web-based infrastructure (such as a fork of Adam Topaz’s server) integrated with Moodle. The student must be able to do mathematics in a single click. The effectiveness of this “zero installation” approach was proven to me by an unexpected experiment:

I had my grandmother test the *Natural Number Game* on her iPad. Although she has no mathematical background, the complete absence of technical barriers allowed her to focus purely on the logic of the game (which she compared to playing Rummikub). The mechanics of the tactics felt very natural to her. This anecdote illustrates a fundamental point: when the cognitive load related to the computing environment is removed, formal logic becomes accessible to anyone.

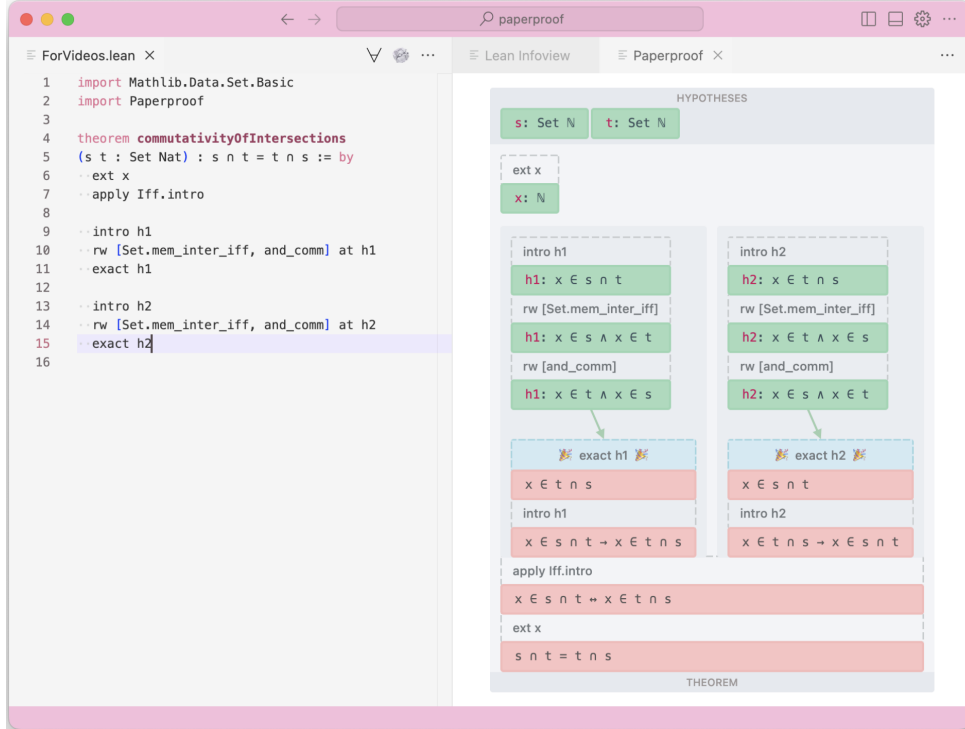


Figure 2: Interactive proof tree generation in Paperproof.

B. The Cognitive Barrier: Preserving Global Intuition with Paperproof. However, even on a simplified Web interface, a second friction point emerges during the proof-writing process: losing the big picture. In Lean, the code is linear, and the student only sees the current state. The risk, echoing Weber’s warning, is getting bogged down in local syntax and losing sight of the global proof tree, making it difficult to bridge the gap between the machine and the conceptual proof on paper.

To mitigate this issue, which I personally experienced, I found visualization tools like **Paperproof** [7] to be particularly well-suited.

By generating an interactive visual graph of the demonstration, it shows exactly how each tactic modifies the proof (creating new branches, altering the goal, etc.).

3.2 The Community Challenge: Sparking Momentum through Events

How can we practically launch this initiative at EPFL and attract students? Integrating Lean should not rely solely on institutional pressure (just another mandatory course), but should first generate genuine enthusiasm. This approach also addresses two practical needs: stress-testing our infrastructure and identifying potential teaching assistants.

A. The Extracurricular Context: Lean as the First Collaborative Mathematical “Product”. At EPFL, mathematical work is traditionally very solitary. While the mathematics student association (CQFD) excels at creating social bonds (notably through organized gatherings), there is no real culture of collaborative projects in mathematics, unlike in computer science. Lean fundamentally changes the game: it transforms a mathematical proof into a “product” on which several students can collaborate in real-time, much like an open-source project. Unlike typical coding events, the ultimate goal remains purely mathematical: formalizing a complete theory together.

B. The Acquisition Model: Drawing Inspiration from the BSA. To build momentum before institutionalizing a course, we can draw inspiration from the funnel model used by the *Blockchain Student Association* (BSA). Their strategy has a proven track record at EPFL:

1. **The Training Phase:** Organize three introductory evenings (informal workshops with pizza). At BSA events, this format easily attracts around a hundred curious students.
2. **The Application Phase:** Conclude this introduction with a Hackathon (which, in the case of the BSA, has successfully gathered up to 250 registrants).

C. A Tailored Hackathon Concept: “Territory Expansion”. A classic Lean hackathon (rewarding the most complex proof) would pose a problem: beginners would immediately be crushed by experts or by the use of artificial intelligence. To bypass this, I propose an alternative concept: a **“Territory Expansion Competition”** (like the Risk board game). Relying on *blueprints*, we could provide a graph of theorems of varying difficulty levels. Over 48 hours, teams would have to conquer as much territory as possible. This format allows beginners to score points for their team by formalizing a multitude of simple lemmas, while leaving the complex theorems to the more advanced participants. Furthermore, the inevitable use of AI can be harnessed by introducing a ‘Quality Metric’ evaluation.

Conclusion

Integrating Lean 4 into the Undergraduate mathematics curriculum at EPFL represents a unique opportunity to fundamentally change how students perceive and practice mathematical rigor. As my own learning data and experience have shown, Lean acts as an incorruptible tutor that shatters the “illusion of rigor” often maintained on paper. However, this pedagogical power comes at a cost: a steep learning curve and a heavy cognitive load caused by syntactic strictness and state management.

To ensure this integration is a success rather than a technical burden for Undergraduate students, the tool itself must fade into the background. This requires a seamless integration, combining a Web-based infrastructure to eliminate installation hurdles, and visual tools like Paperproof to maintain the global mathematical intuition.

Moving forward, the success of this initiative will not rely solely on institutionalizing a new course, but on building a community. To transition from this preliminary study to a full-scale deployment, I propose the following strategic roadmap:

- Deploy the Web-based Lean server integrated with Moodle and visualization tools.
- Partner with student associations to organize introductory workshops followed by the “Territory Expansion” hackathons.
- Use these hackathons as a large-scale stress test for the server infrastructure.
- Identify the most dedicated and high-performing students during these events to serve as future teaching assistants (TAs).

Looking ahead, a natural long-term evolution would be to integrate Lean directly into existing fundamental courses, rather than introducing it as a standalone class. As the limited engagement with optional initiatives like the Topology game suggests, making Lean an integral part of the continuous assessment (requirement / bonus) seems essential if we hope to shape the rigor of the entire cohort.

References

- [1] D. Bessis, *Mathematica: A Secret World of Intuition and Curiosity*, Yale University Press, 2024.
- [2] A. Thoma and P. Iannone, “Learning about Proof with the Theorem Prover LEAN: the Abundant Numbers Task,” *International Journal of Research in Undergraduate Mathematics Education*, vol. 8, no. 1, pp. 64–93, 2021. <https://doi.org/10.1007/s40753-021-00140-1>
- [3] K. Weber, “How Mathematicians Determine if an Argument Is a Rigorous Proof,” *Journal for Research in Mathematics Education*, vol. 39, no. 4, pp. 437–459, 2008.
- [4] B. Århem, *Topology Lean Game* [Interactive learning game for Lean 4], EPFL, 2024. https://adam.math.hhu.de/#/g/bjornarhem/topology_lean_game
- [5] K. Buzzard and M. Pedramfar, *The Natural Number Game* (Lean 4 version), Imperial College London & Lean Community, 2023. <https://adam.math.hhu.de/#/g/leanprover-community/nng4>
- [6] A. Kontorovich, *The Real Analysis Game*, Rutgers University, 2024. <https://adam.math.hhu.de/#/g/alexkontorovich/realanalysisgame/>
- [7] A. Merzlyakov and Paperproof Contributors, *Paperproof: A visual proof assistant and graph visualizer for Lean 4*, 2023. <https://github.com/Paper-Proof/paperproof>
- [8] J. Digard, *Beginner Roadblocks and LLM Logs in Lean 4: Raw Data and Categorization Scripts*, GitHub repository, 2026. <https://github.com/jeremydigard/lean4-learning-friction-analysis>