The Mistrust of Formal Proofs in Pure Mathematics

A Research Paper submitted to the Department of Engineering and Society

Presented to the Faculty of the School of Engineering and Applied Science
University of Virginia • Charlottesville, Virginia

In Partial Fulfillment of the Requirements for the Degree
Bachelor of Science, School of Engineering

Jamie Fulford
Spring 2025

On my honor as a University student, I have neither given nor received unauthorized aid on this assignment as defined by the Honor Guidelines for Thesis-Related Assignments.

Advisor

Caitlin D. Wylie, Department of Engineering and Society

1 Introduction

Software failures in critical systems have resulted in catastrophic consequences. The Therac-25 radiation therapy machine fatally overdosed patients due to concurrency bugs in its control software (Baase, 2012), while the Ariane 5 rocket's self-destruction, triggered by an integer overflow, resulted in a $370 million loss (Dowson, 1997). As software systems increasingly control critical infrastructure–from medical devices to financial systems to autonomous vehicles–preventing such devastating failures has never been more crucial. However, there is a potential solution to this challenge: proof assistants, specialized software tools that help develop and verify formal mathematical proofs of program correctness. As Harrison et al. explain in their comprehensive history of the field, these tools evolved from early automated theorem provers into interactive systems that combine human insight with machine-verified mathematical rigor (Harrison et al., 2014). A formal proof, in this context, is a complete mathematical demonstration of correctness that can be mechanically verified by a computer, providing a level of certainty far beyond traditional testing methods. These powerful tools face a fundamental challenge: while they could revolutionize how we verify both mathematical theorems and software systems, there exists a wide gap between their theoretical potential and practical adoption. Such a gap manifests in both technical barriers to entry and deep social resistance, creating what Kiiskinen describes as a "curiously empty intersection" between proof engineering and computational sciences (Kiiskinen, 2023). This raises a critical question: Why do mathematicians resist adopting proof assistants despite their potential to enhance both mathematical certainty and software reliability?

This paper examines the mistrust of proof assistants within the mathematical community and its implications for software verification and mathematical practice. First, I examine historical case studies and firsthand accounts that demonstrate this mistrust in action, revealing how mathematicians struggle with the translation between intuitive mathematical reasoning and formal verification. Next, I analyze personal experiences with proof formalization that highlight the cognitive and procedural barriers these tools present. Finally, I apply the Social Construction of Technology (SCOT) framework to systematically map the relevant social groups and technological frames that

shape proof assistant development and adoption (Pinch and Bijker, 2012). I argue that this mistrust is not merely resistance to new technology, but reflects fundamental questions about mathematical practice, knowledge validation, and epistemology. By understanding all the social dimensions, we can better address the barriers preventing these powerful verification tools from reaching their full potential in protecting critical infrastructure, ensuring software correctness, and advancing mathematical knowledge.

2 Historical Evidence of Resistance

The mistrust of proof assistants within the mathematical community manifests through direct accounts of mathematicians' experiences with these tools. Koutsoukou-Argyraki documents her journey as an established mathematician learning to use the Isabelle/HOL proof assistant, capturing both technical and psychological barriers mathematicians face when adopting formal verification tools (Koutsoukou-Argyraki, 2021). She compares writing a mathematical proof to creating an elaborate carpet: while the end result appears impressive, the process consists of simple arguments like small stitches. However, when formalizing mathematics with a proof assistant, every such "stitch" must be made explicit—even steps that are trivial to humans must be broken down into elementary sub-steps. Despite her extensive background in mathematical proof theory, she encountered persistent difficulties with Isabelle's rigid syntax and proof construction methods. This disconnect between mathematical intuition and machine formalization represents a fundamental misalignment between how mathematicians conceptualize proofs and how proof assistants require them to be expressed. Her experience was particularly notable given her background in proof mining, which already trained her in deconstructing proofs into elementary sub-steps. Yet even with this relevant experience, the transition to formal verification presented significant challenges, highlighting how deeply these tools conflict with traditional mathematical practice. Her peer-reviewed account provides concrete evidence of the barriers that create resistance to proof assistant adoption within the mathematical community. In particular, while mathematicians might be motivated to

use proof assistants for increased certainty in complex proofs, these fundamental misalignments in how mathematical thinking is expressed create significant barriers to adoption.

This individual experience reflects broader patterns visible in large-scale mathematical formalization projects. Kaliszyk and Urban's analysis of collaborative formalization efforts reveals how these projects become sites of tension between different mathematical practices (Kaliszyk and Urban, 2016). When mathematical communities attempt to formalize established results, they must confront fundamental questions about proof representation and validity that challenge traditional mathematical workflows. The history of the four-color theorem's proof offers a particularly illuminating example of this tension. After more than a century of failed attempts at traditional proof, the theorem was finally proved in 1976 using computer assistance—a breakthrough that was met with significant controversy in the mathematical community (Wilson, 2014). The proof required examining thousands of cases, a task impossible to verify by hand, forcing mathematicians to confront questions about the nature of mathematical understanding and verification. This controversy presaged many of the current debates around proof assistants, as it raised fundamental questions about whether computer-aided proofs provide the same level of mathematical insight as traditional methods. Similar tensions emerged in more recent efforts like the Flyspeck project, which formalized the proof of the Kepler conjecture (Hales et al., 2017). While successfully verifying the proof, the project also highlighted significant cultural resistance from mathematicians who questioned whether machine verification added meaningful mathematical understanding. This resistance stems not from doubts about the logical foundations of proof assistants, but from concerns about losing the intuitive insights that mathematicians value in traditional proof methods. The potential benefits of exhaustive verification must be weighed against what many mathematicians see as a significant cost: the loss of the conceptual clarity and elegance that drives mathematical advancement.

These tensions in how mathematicians view computer-assisted proofs reflect deeper generational and philosophical divides within the mathematical community. Bayer et al.'s analysis of mathematical proof practices across generations reveals how established mathematicians construct

proof validity around concepts of insight and intuition (Bayer et al., 2022). Their work, published as a notice from the American Mathematical Society, demonstrates how traditional mathematical practice values proofs not just as verification tools, but as vehicles for understanding mathematical structures and relationships. This perspective stands in marked contrast to the formal verification approach embodied by proof assistants, which prioritize complete logical rigor over intuitive understanding. Shulman presents a contrasting vision, arguing that proof assistants are actively reshaping what kinds of mathematical questions can be asked and answered, potentially opening new territories for mathematical exploration (Shulman, 2024). However, this optimistic view of proof assistants as tools for mathematical discovery has struggled to gain traction among mathematicians who view computer verification as divorced from genuine mathematical insight. The gap between these perspectives—proof as insight versus proof as verification—represents a fundamental challenge to the adoption of proof assistants in mathematical practice. For mathematicians to embrace proof assistants more widely, these tools would need to demonstrate clear advantages beyond mere verification, such as enabling new discoveries or providing insights that traditional methods cannot, while maintaining the intuitive understanding that mathematicians value.

These historical examples, personal accounts, and generational analyses collectively demonstrate how the mistrust of proof assistants emerges from deep-seated practices and values within mathematical culture. This mistrust manifests across multiple dimensions: from individual mathematicians struggling with formal verification tools, to community-wide controversies over computer-assisted proofs, to fundamental disagreements about the nature of mathematical understanding. While proof assistants have demonstrated their capability for verifying complex mathematical results, the mathematical community's resistance goes beyond simple skepticism of new technology. It reflects a more fundamental concern about preserving the aspects of mathematical practice that have historically driven mathematical discovery. This cultural divide between mathematicians and computer scientists creates a significant challenge for the broader adoption of proof technologies. As critical systems increasingly rely on formal verification for safety, bridging this gap becomes not just an academic concern but a practical necessity for advancing both mathematical practice

and software reliability.

3 Ethnographical and Experiential Evidence

My personal experience with Rocq, a widely-used proof assistant, provides a useful starting point for understanding the dynamics of attitudes toward proof assistants. Despite having both mathematical and programming training, I encountered significant challenges when attempting to formalize basic properties of metric spaces, mathematical structures that define the concept of distance in abstract spaces, a particular mathematical structure I wanted to formalize for a project. The difficulty was not in understanding the mathematics—I could easily prove these properties with pen and paper—but in translating mathematical concepts into a formal language. Defining a metric space in Rocq required numerous attempts as the formal representation repeatedly failed to capture the intuitive mathematical structure I was trying to express. Even expressing a property as fundamental as the triangle inequality became an exercise in formal manipulation rather than mathematical reasoning. The most frustrating aspect was realizing that established mathematical notation, which elegantly captures core ideas, had to be abandoned for verbose machine-readable definitions that obscured the underlying mathematical intuition. After finally establishing the basic definitions, I discovered my formalization was incompatible with existing libraries that used different representation choices. This experience revealed how proof assistants force mathematicians to make formal commitments to representation choices that mathematical practice deliberately leaves flexible, creating what Koutsoukou-Argyraki identifies as a fundamental ”translation barrier” between mathematical thinking and formal verification systems (Koutsoukou-Argyraki, 2021).

The tactical approach of proof assistants presents another significant barrier to mathematical thinking. After establishing basic definitions in Rocq, I faced the challenge of constructing formal proofs using its tactic language. Tactics are commands that manipulate proof states—essentially small programs that automate proof steps that would be considered ”obvious” in mathematical practice. While tactics ostensibly simplify proof construction, they introduce a separate mental

model that bears little resemblance to intuitive mathematical reasoning. For example, when proving a certain property about metric spaces, I found myself spending hours learning tactical patterns rather than focusing on the mathematical content. The proof script ultimately resembled a programming solution rather than a mathematical argument, with nested tactical combinations that obscured the elegant mathematical insight the proof was meant to convey. As I gained more experience with tactics like ”destruct,” ”rewrite,” and ”apply,” I realized I was developing proficiency in Rocq itself rather than deepening my understanding of the mathematics. This tactical layer creates what I would call a ”procedural barrier” that compounds the translation difficulties identified earlier. The resulting disconnect between tactical manipulation and mathematical insight helps explain why formal verification often feels like an exercise in programming rather than mathematical reasoning, lacking the conceptual clarity that makes traditional proofs valuable as tools for understanding.

These personal experiences illuminate a deeper pattern that connects to broader scholarly discussions about the accessibility of proof assistants. My struggles with formal representation echo what Volker identified as a fundamental design problem in proof assistant interfaces. As he observed, proof assistants have been primarily developed by ”academics or students” with limited incentives to prioritize usability (Völker, 2004, p. 140). This observation provides critical context for understanding why the experience of working with proof assistants feels so disconnected from traditional mathematical practice. The tactical approach to proof construction presents particularly revealing challenges that go beyond mere interface design. Blanchette et al.’s analysis of ”hammer” systems—tools designed to automate parts of the proof process—attempts to address this tactical complexity (Blanchette et al., 2016). Their work on automating proof tactics demonstrates recognition within the proof assistant community that tactical complexity represents a significant barrier. However, their focus remains primarily on technical solutions to automation rather than addressing the cognitive mismatch between tactical thinking and mathematical reasoning. This gap between technical capability and mathematical practice reveals that even as proof assistants advance technically, they continue to diverge from the cognitive processes that make mathematics valuable as

a form of reasoning. This analysis points to a more substantial conclusion: the resistance to proof assistants within mathematics may represent not just conservative reluctance to adopt new technology, but a legitimate defense of valuable cognitive practices that formal verification currently disrupts. Shulman's optimistic vision of proof assistants opening "strange new universes" of mathematical possibility (Shulman, 2024) may eventually prove correct, but my experience suggests that realizing this potential will require fundamental rethinking of how proof assistants engage with mathematical cognition, not just incremental improvements in their technical capabilities. As infrastructure increasingly relies on software systems that could benefit from formal verification, the barriers preventing mathematicians from engaging with proof assistants contribute to persistent verification gaps. These gaps leave critical systems vulnerable to the same kinds of failures that caused the Therac-25 radiation overdoses and Ariane 5 explosion referenced earlier.

A mathematician and software developer who has contributed extensively to Lean's mathlib offers a compelling counterargument to the experiences described above (Conneen, 2025). They contend that well-designed formal libraries like mathlib are deliberately constructed to be explanatory, not just functional. The code structure in mathlib mirrors the conceptual structure of the mathematics being formalized, with definitions and lemmas (smaller supporting theorems) organized to reflect mathematical relationships. Bayer et al. support this perspective when they note that younger mathematicians increasingly view computer-assisted proofs as "natural extensions of mathematical reasoning" (Bayer et al., 2022). This suggests that the disconnection experienced by many mathematicians might result from insufficient training rather than inherent limitations of the technology. However, this argument does not fully address the fundamental cognitive shift required to translate between mathematical intuition and formal verification. Even if mathlib's structure is mathematically motivated, engaging with it requires mathematicians to adopt computational thinking patterns that may disrupt their mathematical creativity.

Terence Tao's advocacy for proof assistants, particularly through his blog post on the Lean4 formalization of the PFR theorem (the Prime Formulation of the Riemann Hypothesis, a significant result in analytic number theory), highlights both the promise and the friction involved in

integrating formal verification into mathematical practice (Tao, 2023). Unlike those who present proof assistants as a straightforward enhancement to mathematical rigor, Tao's approach implicitly acknowledges the fundamental cognitive shift they require. His detailed breakdown of the formalization process reveals how translating mathematical intuition into machine-verifiable syntax introduces layers of complexity that many mathematicians find counterintuitive. While he demonstrates enthusiasm for the Lean4 project, his documentation of struggles—such as reconciling formal definitions with existing mathematical structures or managing the procedural constraints of Lean's tactics—offers an unfiltered look at the intellectual challenges of proof formalization. Tao's engagement with these difficulties suggests that the integration of proof assistants is not merely a technical endeavor but a transformation of mathematical reasoning itself. By inviting open collaboration through platforms like GitHub and Zulip, he encourages a shift toward a more collective and iterative approach to theorem verification. However, his work also raises the question of whether proof assistants can evolve in ways that accommodate, rather than disrupt, the intuitive modes of mathematical thought that have long defined the discipline.

4 Theoretical Analysis

The Social Construction of Technology (SCOT) framework, which analyzes how social factors influence technological development, provides a systematic way to analyze how different groups have shaped and interpreted proof assistants. Developed by Pinch and Bijker, SCOT rejects technological determinism—the idea that technologies evolve according to an internal logic—and instead focuses on how social processes shape technological development (Pinch and Bijker, 2012). While it might be tempting to view the adoption of proof assistants as a simple division between mathematicians and computer scientists, Harrison et al.'s historical analysis reveals a more complex landscape of social groups (Harrison et al., 2014). Traditional pure mathematicians, who value proofs primarily for their insight and intuition, represent one interpretive framework. Computer scientists working in formal methods form another distinct group, viewing proof assistants as natu-

ral extensions of computational logic. Software verification specialists constitute a third significant group, focused on practical applications in critical systems. A fourth key group emerged more recently: younger mathematicians who, as documented by Koutsoukou-Argyraki, approach these tools with different expectations shaped by contemporary computational practices (Koutsoukou-Argyraki, 2021). Each of these groups has constructed fundamentally different understandings of what proof assistants are and what role they should play in mathematical practice, leading to what SCOT terms "interpretive flexibility" (Pinch and Bijker, 2012)—the way that different social groups can develop radically different understandings of the same technology.

The development trajectory of proof assistants has been significantly shaped by academic power structures and incentives, influencing how different groups interact with these tools. Volker's analysis reveals how academic incentives have historically privileged certain interpretations of proof assistants over others (Völker, 2004). The primary developers of these tools have been computer scientists in academic settings, who face little external pressure to accommodate the needs of other groups. As Volker notes, "there is little external reward for building and maintaining such user interfaces... most developers are academics or students" (Völker, 2004, p. 140). This academic context has created what SCOT would term a "technological frame" that prioritizes theoretical completeness over practical usability, fundamentally affecting how proof assistants have evolved (Pinch and Bijker, 2012). The resulting tools naturally align with the computer science perspective of formal verification, while potentially alienating other groups—particularly traditional mathematicians—who might conceptualize proofs differently. This disparity in influence over tool development has reinforced existing power dynamics between different social groups, contributing to the ongoing tensions around proof assistant adoption.

Blanchette et al.'s work on "hammer" systems provides an illuminating case study of how technological solutions emerge within specific technological frames (Blanchette et al., 2016). Hammers are tools designed to automate parts of the proof process by connecting interactive theorem provers to automated provers. Within the SCOT framework, hammers represent an attempt to address usability challenges, but they still emerge primarily from the formal methods technological

frame (Pinch and Bijker, 2012). While hammers aim to reduce the tactical complexity that many users struggle with, they don't address the fundamental disconnect between mathematical thinking and formal verification. The automation hammers provide doesn't preserve the mathematical intuition and insight that mathematicians value most in proofs. Instead, they accelerate the formal verification process while still requiring users to translate mathematical concepts into formal specifications. This case illustrates how technological solutions often address symptoms rather than root causes when they emerge from within a single technological frame without sufficient input from other relevant social groups.

What SCOT reveals about proof assistants suggests a path forward that goes beyond technical improvements alone. Achieving what SCOT calls "rhetorical closure"—where controversies around a technology appear resolved—would require creating spaces of genuine dialogue between the different social groups identified above. This means moving beyond interdisciplinary conferences dominated by computer scientists to forums where mathematicians have equal voice in defining what proof assistants should be. A form of "redefinition of the problem" closure might be necessary, where proof assistants are reconceptualized not as replacements for traditional mathematical reasoning but as complementary tools that serve different purposes. This reframing would acknowledge the value of both intuitive mathematical insight and formal verification rather than positioning them as competing approaches. The "inclusion/exclusion" aspect of SCOT suggests that broadening participation in proof assistant development beyond academic computer scientists could significantly reshape these tools. Involving mathematicians earlier in design processes could help develop interfaces and interaction models that better reflect mathematical cognition. These social interventions, rather than merely technical improvements, offer the most promising path toward realizing the potential of formal verification for protecting critical infrastructure while preserving the valuable aspects of mathematical thinking that drive discovery.

5 Conclusion

The mistrust of proof assistants in mathematics represents a sociotechnical challenge with significant implications beyond academia. This analysis has demonstrated that this mistrust stems not from simple resistance to technology, but from fundamental misalignments between mathematical cognition and formal verification approaches. The gap between mathematical intuition and machine-verifiable syntax creates cognitive barriers that many mathematicians find difficult to justify. As critical infrastructure increasingly relies on software systems, these barriers to formal verification contribute to persistent verification gaps that leave essential systems vulnerable to the same kinds of catastrophic failures exemplified by the Therac-25 and Ariane 5 incidents.

The SCOT framework reveals that addressing this challenge requires both technical innovation and social interventions. Proof assistant development has been shaped by technological frames that prioritize formal methods perspectives over mathematical practice, creating tools that align with computer science values rather than mathematical ones. Moving forward demands creating genuine dialogue between stakeholder groups, reconceptualizing proof assistants as complementary to traditional mathematical reasoning, and involving mathematicians as co-creators rather than merely end-users. As our dependence on software systems grows, reconciling mathematical intuition with formal verification becomes increasingly urgent—not just for advancing mathematical knowledge, but for ensuring the safety and reliability of the software systems that underpin contemporary society.

References

Baase, S. (2012, August). *A gift of fire: Social, legal, and ethical issues for computing technology* (4th ed.). Pearson Prentice Hall.

Bayer, J., Benzmuller, C., Buzzard, K., David, M., Lamport, L., Matiyasevich, Y. V., Paulson, L. C., Schleicher, D., Stock, B., & Zelmanov, E. I. (2022). Mathematical proof between generations. *ArXiv*, *abs/2207.04779*. https://api.semanticscholar.org/CorpusID:250426455

Blanchette, J. C., Kaliszyk, C., Paulson, L. C., & Urban, J. (2016). Hammering towards qed. *Journal of Formalized Reasoning*, *9*(1), 101–148. https://doi.org/10.6092/issn.1972-5787/4593

Conneen, C. (2025). Personal interview [Interview conducted in person]. https://topos.place

Dowson, M. (1997). The ariane 5 software failure. *SIGSOFT Softw. Eng. Notes*, *22*(2), 84. https://doi.org/10.1145/251880.251992

Hales, T., ADAMS, M., BAUER, G., DANG, T. D., HARRISON, J., HOANG, L. T., KALISZYK, C., MAGRON, V., MCLAUGHLIN, S., NGUYEN, T. T., & et al. (2017). A formal proof of the kepler conjecture. *Forum of Mathematics, Pi*, *5*, e2. https://doi.org/10.1017/fmp.2017.1

Harrison, J., Urban, J., & Wiedijk, F. (2014). History of interactive theorem proving. In J. H. Siekmann (Ed.), *Computational logic* (pp. 135–214, Vol. 9). North-Holland. https://doi.org/10.1016/B978-0-444-51624-4.50004-6

Kaliszyk, C., & Urban, J. (2016). Wikis and collaborative systems for large formal mathematics. In P. Molli, J. G. Breslin, & M.-E. Vidal (Eds.), *Semantic web collaborative spaces* (pp. 35–52). Springer International Publishing.

Kiiskinen, S. (2023). Curiously empty intersection of proof engineering and computational sciences. In P. Neittaanmäki & M.-L. Rantalainen (Eds.), *Impact of scientific computing on science and society* (pp. 45–73). Springer International Publishing. https://doi.org/10.1007/978-3-031-29082-4_3

Koutsoukou-Argyraki, A. (2021). Formalising mathematics – in praxis; a mathematician's first experiences with isabelle/hol and the why and how of getting started. *Jahresbericht der Deutschen Mathematiker-Vereinigung*, *123*(1), 3–26. https://doi.org/10.1365/s13291-020-00221-1

Pinch, T., & Bijker, W. (2012). *The social construction of technological systems: New directions in the sociology and history of technology*. The MIT Press. Retrieved November 9, 2024, from http://www.jstor.org/stable/j.ctt5vjrsq

Shulman, M. (2024). Strange new universes: Proof assistants and synthetic foundations [Published electronically: February 15, 2024]. *Bulletin of the American Mathematical Society*, *61*, 257–270. https://doi.org/10.1090/bull/1830

Tao, T. (2023, November). Formalizing the proof of pfr in lean4 using blueprint: A short tour. https://terrytao.wordpress.com/2023/11/18/formalizing-the-proof-of-pfr-in-lean4-using-blueprint-a-short-tour/

Völker, N. (2004). Thoughts on requirements and design issues of user interfaces for proof assistants [Proceedings of the User Interfaces for Theorem Provers Workshop, UITP 2003]. *Electronic Notes in Theoretical Computer Science*, *103*, 139–159. https://doi.org/10.1016/j.entcs.2004.05.001

Wilson, R. (2014). *Four colors suffice: How the map problem was solved - revised color edition* (Vol. 128). Princeton University Press. Retrieved January 22, 2025, from http://www.jstor.org/stable/j.ctv1vbd2gs