

BEST OF

Books & Arts in Review

11 Books We Read This Week

The Best Books of September

September in Fiction

Sum

This copy is for your personal, non-commercial use only. Distribution and use of this material are governed by our Subscriber Agreement and by copyright law. For non-personal use or to order multiple copies, please contact Dow Jones Reprints at 1-800-843-0008 or visit www.djreprints.com.

<https://www.wsj.com/arts-culture/books/the-proof-in-the-code-review-lean-mean-computing-machine-3fd9847e>

ARTS & CULTURE • BOOKS • BOOKSHELF

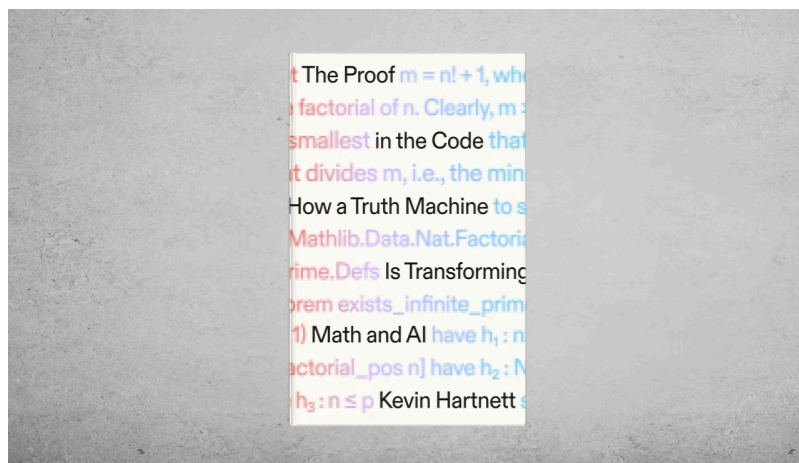
Follow

'The Proof in the Code' Review: Lean, Mean Computing Machine

The program called Lean was built to detect bugs in Microsoft's products. It ended up revolutionizing mathematics.

By Amir Alexander

Oct. 2, 2026 10:38 am ET



The mathematical world was shaken to its core last month, when OpenAI announced that it had solved one of the field's most celebrated problems. The Navier-Stokes problem, which deals with complex fluid dynamics, had withstood the assaults of mathematicians for almost two centuries. To tackle it OpenAI unleashed a swarm of 10,000 artificial-intelligence agents, which took 88 hours to arrive at the result and an additional 17 hours to confirm it via the computerized verification program Lean. Along the way, the company scooped New York University mathematician Tristan Buckmaster and his collaborators, who were closing in on a solution. The announcement was a shock to many mathematicians and generated much soul-searching. But it may not have surprised Alan Turing.

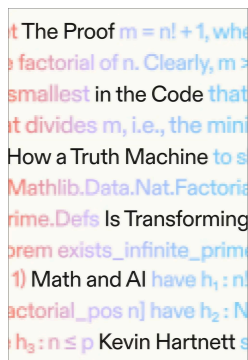
 GRAB A COPY

The Proof in the Code: How a Truth Machine Is Transforming Math and AI

By Kevin Hartnett

Quanta Books

288 pages



Turing was only 24 in 1936, when he came up with a simple but radical thought experiment: Imagine a machine, he proposed, made up of an infinite tape divided into cells with signs written in them, and a “head” (his term) that can read and write signs. The head would read the sign in a cell, respond by writing another sign, and then move to the next cell and do the same—all depending on a pre-existing set of rules. The

proposal appeared both trivial and pointless: After all, the machine doesn’t “do” anything except change meaningless signs into other meaningless signs. Yet the “Turing machine,” as it came to be known, provided the theoretical foundations for all computers to this day.

Turing was not a computer scientist, even if he invented the field. He was a mathematician, and his machine was the embodiment of “formalism,” an idea that was then transforming his field—that mathematics, far from being the study of universal truths, is nothing but the manipulation of meaningless signs according to predetermined rules. What, then, could be more appropriate than a machine that does exactly that? The Turing machine, it followed, could produce all possible mathematics and do it far better than an error-prone human.

As Kevin Hartnett relates in “The Proof in the Code,” soon after digital computers became available their creators attempted to put Turing’s proposal to the test. In the 1950s and ’60s, researchers developed Automated Theorem Provers that could examine long mathematical formulas to determine whether they were true under certain conditions. ATPs proved powerful and effective—at least for certain mathematical problems.

It became clear, however, that most questions of interest to mathematicians are not so easily put to a machine. And so, compromising on the ideal of mechanized proof, researchers developed Interactive Theorem Provers: Mathematicians propose a formal argument, and the program checks whether it is correct and under what conditions. Using this more flexible tool, Kenneth Appel and Wolfgang Haken succeeded in 1976 in proving the Four Color theorem that had stood for more than a century, and Thomas Hales did the same for an even more august problem, the four-century-old Kepler conjecture.

Even so, ITPs remained unpopular with practicing mathematicians. They quickly discovered that making a mathematical argument understandable to a computer program required an enormous amount of work. The argument had to be reduced to a mind-numbing step-by-step formal argument, and so did all the mathematics that it was based on. In normal communications mathematicians rely on an extensive body of knowledge that they can assume their colleagues are familiar with. To present an argument to an ITP, all this background knowledge must be entered, explicitly and formally, down to the simplest definitions of numbers. Paradoxically, trying to solve a mathematical problem with the aid of computers often turned out to be a lot more work than handling it the traditional way.

“The Proof in the Code” tells the story of Lean, the ITP that finally made proof by computer central to mainstream mathematics. Lean was the brainchild of Leonardo de Moura, a computer scientist at Microsoft Research in the early 2000s. Mr. de Moura’s initial goal was mundane: to create a computer program that would find hidden bugs in Microsoft’s products. His Z3 program proved highly effective in doing so for Windows 7 before its 2009 release, and Lean was designed to go one better: Whereas Z3 could only check particular runs of a program with particular values, Lean was designed to look for bugs in programs as a whole.

Mr. de Moura may have intended Lean to serve Microsoft’s bottom line, but he soon discovered that the people most interested in the program were mathematicians. Testing a computer program for bugs, it turns out, is

almost indistinguishable from checking a formal mathematical proof for possible errors—which is precisely what ITPs do. And so, Mr. de Moura began working closely with a coterie of mathematicians intent on making computers a standard tool in their field.

Mr. Hartnett is at his best when describing the different personalities and dynamics in this small club. Mr. de Moura, brilliant but unassuming, keeps the project on track through sheer doggedness. His closest collaborator, Jeremy Avigad, is a classic academic mentor who recruits his students to the cause. Mario Carneiro, a younger man who sometimes clashes with Mr. de Moura, possesses an unquenchable passion for formalizing proofs. Kevin Buzzard, a flamboyant British number theorist, believes that Lean will cleanse mathematics of its excessive sloppiness.

To eliminate the Sisyphean task of formalizing all relevant mathematics from scratch every time, the Lean enthusiasts decided to create Mathlib—a library of formalized mathematics. This was a long, open-ended undertaking that caused friction in the team. In particular Mr. de Moura, who was focused on improving Lean's core capabilities, spent long hours each day correcting Mathlib entries made by other contributors, who seemed to him demanding and disrespectful. But over time, as the library grew and news of it spread, more mathematicians were drawn to contribute to and join the Lean circle.

A breakthrough came in late 2020. Peter Scholze, a winner of the Fields Medal (known as the “Math Nobel”), asked Mr. Buzzard for help: Could Lean verify his latest and most complex proof, in a field known as “liquid tensors”? Since the proof relied on a mountain of previous results, running it through Lean required formalizing all that mathematical knowledge and entering it into Mathlib. The project involved 28 mathematicians and 18 months of work, but by July 2022 Mr. Scholze announced Lean had verified his proof. A year later Terence Tao of the University of California, Los Angeles—another Fields medalist and perhaps the most influential mathematician in the world—recruited an even larger group to use Lean to verify his proof of the polynomial Freiman-Ruzsa conjecture. There could no longer be any doubt: Lean had become a pillar of advanced mathematics.

But even as Lean was establishing its worth, another revolution was taking place outside the halls of academia. First OpenAI and then numerous other companies began offering large language models that could write prose, translate speech, produce videos on command and more. Trained on stupendous amounts of human-generated sources, these LLMs use statistical algorithms to produce results that are all but indistinguishable from human creations.

Initially LLMs were terrible at math. When asked to generate a proof, they would produce a text that looked like a proof but was logically incoherent. Then Thomas Hubert of Google's DeepMind came up with a way to improve their performance: Instead of settling for a bad "proof," LLMs would enter their initial work product into Lean, which would provide feedback. The LLM would use this feedback to produce a better version that it would again enter into Lean. Over many iterations, the AI engine could produce a viable proof. This cycle, multiplied many times over by the use of autonomous AI agents, was essential to OpenAI's Navier-Stokes result.

Have we finally arrived at a true Turing machine that can produce advanced math without human input? There is a good argument to be made that we have. Yet reading "The Proof in the Code," which takes the reader right up to the threshold of current breakthroughs, I was struck as much by the limitations of machine mathematics as by its power. Lean can verify any sequence of deductions, but it still relies on Mathlib—a human-created aggregate of mathematics that humans found important and meaningful.

Similarly, OpenAI's proof of Navier-Stokes was without question a fantastically complex exercise in sign manipulation. But it was humans who decided that spending millions of dollars on AI agents to pursue this particular sign-manipulation exercise was important. So we must ask: Could a method of proof that excludes human insight, and must be accepted on faith, be considered mathematics? In a recent letter, more than 25 Fields medalists (including Messrs. Scholze and Tao) answered: "no." OpenAI's proof is almost certainly correct. But a result that does nothing to advance human understanding, they contend, can hardly be called mathematics.

The field is at a turning point. The mainstreaming of Lean and the inroads made by LLMs have forced mathematicians for the first time in more than a century to reckon with the fundamentals of their field. What counts as mathematics? What counts as a proof? And who, human or machine, should get the credit? Kevin Hartnett's "The Proof in the Code" is a lively and very human account of how we got here.

—Mr. Alexander teaches the history of science and mathematics at the University of California, Los Angeles. His most recent book is "Liberty's Grid: A Founding Father, a Mathematical Dreamland, and the Shaping of America."

Appeared in the October 3, 2026, print edition as 'A Lean, Mean Math Machine'.

Videos

