Mathematical Proof Between Generations

Jonas Bayer, Christoph Benzmüller, Kevin Buzzard, Marco David, Leslie Lamport, Yuri Matiyasevich, Lawrence Paulson, Dierk Schleicher, Benedikt Stock, and Efim Zelmanov

A *proof* is one of the most important concepts of mathematics. However, there is a striking difference between how a proof is defined in theory and how it is used in practice. This puts the unique status of mathematics as exact science into peril. Now may be the time to reconcile theory and practice, i.e., precision and intuition, through the advent of *computer proof assistants*. This used to be a topic for experts in specialized communities. However, mathematical proofs have become increasingly sophisticated, stretching the boundaries of what is humanly comprehensible, so that leading mathematicians have asked for formal verification of their proofs. At the same time, major theorems in mathematics have recently been computer-verified by people from outside of these communities, even by beginning students. This article investigates the different definitions of a proof, the gap between them, and possibilities to build bridges. It is written as a *polemic* or a *collage* by different members of the communities in mathematics and computer science at different stages of their careers, challenging well-known preconceptions and exploring new perspectives.

DOI: https://doi.org/10.1090/noti2860

1. What is a Mathematical Proof?

Dierk Schleicher

Mathematics often prides itself as the most fundamental of all sciences: a mathematical truth, once established, will be true forever. But what exactly is a mathematical proof? Here is one possible answer in terms of the other fundamental concept in mathematics, a definition:

Definition (Proof: formal definition)**.** A mathematical proof is a sequence of deductions that are based on a given set of axioms and formally deduce consequences following formal rules of deduction.

This might be the first, "idealistic," definition of proof that one encounters, and it is not too hard to prove some simple results in this formal way, perhaps in combinatorics, elementary number theory, or on Euclidean triangles. But sooner rather than later, one discovers that this definition is too clumsy and impractical for any profound result in mathematics. In practice, mathematicians use a very different definition of proof.

Definition (Proof: practical definition)**.** A mathematical proof is a sequence of arguments that convinces an educated reader.

Dierk Schleicher is a professor of mathematics at Aix-Marseille University. His email address is `dierk.schleicher@univ-amu.fr`.

So one might object that mathematicians routinely make one of the most fundamental mistakes: define things in one way and then use them in a different way. But does this not give away all the fundamental virtues of mathematics?

Many mathematicians are quick to point out that a typical proof, given as a convincing sequence of arguments, can be elaborated in more detail: in principle, for every step one may ask why this is so, and one can insert additional steps of explanation. Each of these steps, in turn, may be expanded if need be, until eventually one arrives at the axioms. One would expect that any of these expansion steps can be performed by the author of a proof, until one arrives at sufficiently fundamental levels when one reaches statements that are already established. It is a common understanding among mathematicians that most current mathematical knowledge can be thus justified—in principle. One may thus reinterpret the practical definition of a proof in the sense that a proof is a sequence of arguments that convinces an educated reader that a formal proof in the sense of the first definition "can be constructed."

But it is true that mathematical "proofs" written by humans occasionally do have errors. Many of them are of a trivial kind when some missing steps can be inserted, or an overlooked special case can be treated in the same way as the rest of the proof. Sooner or later, one is often told, any mistakes would be found by the scrutiny of the mathematical community.

But now and again there are theorems that were thought to be proven where much later the proof is recognized to be false or incomplete, and where perhaps even the result itself is recognized as wrong. How would you know that for a given theorem the "sooner or later" might not happen tomorrow, when a possibly fundamental flaw is discovered? How can one then be certain of the mathematical correctness of a proof?

A rather prominent classical example is the four-color theorem: this theorem was thought to be proved already in the late 19th century, when not one but two proofs were given—that were both found to be incorrect eleven years later.

A recent example from dynamical systems. Proofs that are later discovered to be flawed are not just a theoretical possibility, or an anecdote from the early times of mathematics (when perhaps standards of mathematical rigor were not so high). This is illustrated in a very current example from dynamical systems. The claim is stated quite easily:

Let $f : \mathbb{C} \to \mathbb{C}$ be a holomorphic function but not a polynomial (i.e., a transcendental entire function). Then f can have at most one maximal completely invariant domain: that is a domain U with $f(U) = U = f^{-1}(U)$, and such that $\mathbb{C} \setminus U$ contains at least two points; maximality means that U is not contained in a strictly larger domain U' with the same properties. Until very recently, this result would be stated as a theorem, proved by Noel Baker, an eminent pioneer in the field, in 1970 [1]. However, Duval observed about a half century later that the proof is flawed, and Rempe and Sixsmith [19] showed that it cannot be fixed by the methods stated. In particular, a key step in the proof is the statement that if $U, V \subset \mathbb{C}$ are disjoint simply connected domains, then one of $f^{-1}(U)$ and $f^{-1}(V)$ must be disconnected. This statement, however, is false; even as simple a function as $f(z) = e^z + z$ provides a counterexample, as shown by Rempe and Sixsmith: it has infinitely many disjoint simply connected domains with connected preimages. Baker's proof thus cannot be repaired along the lines of its original version. The main result, as of today, is an open question—half a century after it was accepted as a correctly proved theorem.

One might wonder how relevant this result is—perhaps the error went unnoticed because nobody cared? Unfortunately, this is not so. Rempe and Sixsmith [19, Section 9] give a shocking list of results that depend on the flawed paper in a variety of ways. One list contains several publications by a variety of authors that all depend either on the flawed paper, or even on other papers that used the flawed result, but where the main results can be fixed by other methods developed in [19]. Another list contains several publications, some of them 35 years old, that use the flawed paper and for which the main results must now be considered open once again. Yet another list contains two survey papers, between 25 and 30 years old, that refer to the result in [1]. And perhaps most alarmingly, there are several publications, some of which meanwhile classical and much-cited, where the flawed method of proof was found so useful that it was adapted, further developed, and generalized (without recognizing the flaw). As a result, a whole area of mathematics has had to sort out how its theory was affected by a rather "convincing" mistake made half a century earlier.

This is just a recent example, rather worrisome for the mathematical community, that shows that problems can be discovered at any time even in supposedly well-known results. However, it may well be that mathematics is now developing toward the point when, finally, the two definitions of "proof" can be reconciled. How this may come about is one of the key topics of this paper.

Another development in current mathematics, leading to similar conclusions, is that proofs tend to become longer and more complex, possibly so much so that they can no longer be stored as a whole in human memory, nor be verified by referees. For instance, Peter Scholze has recently asked that one of his key results on condensed mathematics be formally verified, a project called the *Liquid Tensor Experiment* that has attracted broad attention and was successfully completed in 2022. A number of years earlier, Vladimir Voevodsky had similarly been worried about the correctness of results in his field leading to the development of *Homotopy Type Theory*.

One of the inspirations for this paper was a workshop at the *Heidelberg Laureate Forum* in 2018, titled "The future of mathematical proof," where many of the coauthors of this paper were present, and where some of the aspects treated here were discussed and developed.

This paper should be seen as a kaleidoscope of many aspects of mathematical proofs and computer proof assistants, contributed by various researchers with diverse backgrounds. In Section 2, *Yuri Matyiasevich* describes his vision that, soon, mathematical publications need to be supported by formal proofs, and that this may well be accomplished by the coming generation. In Section 3, *Efim Zelmanov* argues in favor of the meaning and explanations of mathematics, rather than formal verification. In Section 4, *Leslie Lamport* introduces a way of presenting mathematical proofs that "should make it much harder to publish false proofs." In Section 5, *Christoph Benzmüller* discusses several major accomplishments of proof assistants and derives a vision for integrated formal and traditional proofs. Then, in Section 6, *Jonas Bayer, Marco David, and Benedikt Stock*, three undergraduate students at the time, describe how they learned to work with proof assistants from early on, and completed one of the first major formalizations done by the coming generation. In Section 7, *Kevin Buzzard* explains how he makes proof assistants "sexy" to mathematicians and recounts how he incorporated them into classroom teaching. In Section 8, *Lawrence Paulson* outlines open challenges and a perspective for the future.

Our contributors express different points of view in this text, so it is natural that upon reading, you might not agree with all of their claims. In fact, this also applies to contributors and referees: reading some of the exchanged messages led one of us to remark, "is this still mathematics or already real-life soap opera?" We hope that the kaleidoscope in this text will provide inspiration and help readers develop their own point of view.

Dierk Schleicher

2. Why Formalize Mathematical Results? And Why Hilbert's Tenth Problem?

Yuri Matiyasevich

A decade ago, the St. Petersburg mathematical society held a meeting titled "Mathematical proof: yesterday, today, tomorrow." Being one of the three spokesmen, I completely disagreed with the previous speaker, and dared [15] to publicly announce the following:

> PREDICTION
>
> In 25 years mathematical journals (if they are destined to survive so long) won't take into consideration any paper unless it is accompanied by proofs which can be verified by a computer.

Already at that time at least one (and only one?) journal existed in which proofs passed preliminary computer verification. This was *Formalized Mathematics* founded in 1990. Over the past 10 years, mathematical journals continued to multiply in number but I do not know of any new journal requiring formalized proofs. Nevertheless, I dare to repeat my prediction *verbatim et litteratim*, that is meaning "In 25 years *from now*. . . ."

In fact, the progress in computer verification of proofs is very impressive, both in the software development and in the number and the significance of actually verified theorems. There exists a compendium [23] of such achievements; the list of proofs is regularly updated but initially restricted by a selection of "Top 100" theorems.

Yuri Matiyasevich is a mathematician at the Steklov Institute of Mathematics at St. Petersburg. His email address is `yumat@pdmi.ras.ru`.

But what could be the goal of computer verification of results already proven by human beings? One obvious answer is as follows: to get additional trust in the correctness of a proof (there are many examples of important and widely used theorems with flaws in their proofs which remained unrevealed for decades). However, some people believe that computers themselves are not sufficiently reliable because of possible errors in their design/software/runs. In my opinion, "in 25 years" the progress in this area will remove such objections completely.

Formalized proofs are vital for the very ambitious project of a *World digital mathematics library* [11]. The ultimate goal of this project is to transfer all *mathematical knowledge* (not just axioms/definitions/theorems/proofs) to computers. This would require tremendous efforts but who would care to spend her/his precious time on the meticulous presentation of known results? Mathematicians prefer to produce new ideas (definitions, hypotheses, theorems) and do not appreciate the hard work of writing proofs with all minute details (luckily, computer scientists do appreciate such occupation).

The following remedy was proposed a long time ago: senior, "retired" mathematicians, who are no longer capable of generating new brilliant mathematical ideas, could devote the rest of their lives to "teaching mathematics to computers." But there is an opposed option: this could be done by the young people who just start to master mathematics. In this way they can get an acute feeling of what mathematical rigor is. However, the following remains uninvestigated: how would involvement in such an activity influence the ability to create new mathematics?

So when I heard that a group of students from Jacobs University in Bremen had studied, under supervision of Professor Dierk Schleicher, the proof of the undecidability of Hilbert's tenth problem, I suggested to them to demonstrate their understanding of the whole construction by producing a fully formalized proof. My role in the project was very restricted: I supplied the students with a sufficiently (for human beings) detailed proof, and I was responsible for the choice of *Isabelle* as the verifier. I was happy to see that the students became very enthusiastic, maybe because they (and I too) at that moment underestimated the amount of required work.

Hilbert's tenth problem is not very difficult for formalization. It is a bit strange that at first we were waiting for it for half a century but then four independent verifications in Coq, Isabelle/HOL, Lean and Mizar emerged in a short time ([17], [16], [21], [14]).

Yuri Matiyasevich

3. On Proof and Progress in Mathematics: From the Perspective of a Research Mathematician

Efim Zelmanov

I will add my 5 cents to the wonderful discussion of computer proof verification, organized by students. For more than 40 years, I've lived in the world of proofs and, sometimes, complicated proofs.

If I were told that a proof is correct because a computer program says so, but I don't see big ideas "turning the wheels," then probably I would continue thinking about the problem, as if a computer blessing did not exist.

The purpose of a proof is *understanding*. For mathematicians it is not enough to know if this or that statement is correct or not. They want to know *why* it is correct or not. Often this understanding comes as a link to some big ideas that come into play now and then in different contexts. I am not able to say it better than W. Thurston in his beautiful paper on proofs (see [22]).

In my opinion, a credible computer verification of a proof is an amazing achievement in AI. It is valuable and interesting for its own sake, leaving alone proofs. It may also find other applications.

I have to admit that as far as a straightforward computation is concerned, I trust computers more than humans.

Difficult and important proofs are often written at the edge of human intellectual abilities (think of the proof of G. Perelman of the Poincaré conjecture). Should we expect the authors to present it in a computer-friendly form? I am afraid that it is too much to ask.

Efim Zelmanov is Chair Professor at the Southern University of Science and Technology. His email address is `efim.zelmanov@gmail.com`.

Modern mathematics created a new challenge: proofs of enormous complexity. I know proofs of some far-reaching statements that have been around for quite a while and still nobody (except the authors) can say that they understand all the details. The only hope is that good proofs are like live organisms. They evolve with time and go through a natural selection. New ideas appear and bring new understanding. And new proofs may appear.

I will end with the controversial statement that in practice a proof is what is considered to be a proof by all mathematicians.

Efim Zelmanov

4. Making Math More Rigorous

Leslie Lamport

Mathematics, as practiced by most mathematicians, is not very rigorous. There is evidence that about 1/3 of all published, refereed math papers contain significant errors—incorrect proofs or theorems that their authors believed to be correct. (I have presented evidence elsewhere [12].) Math can be made more rigorous, and mathematicians can make fewer errors, by replacing archaic customs with more sensible practices. Here is how.

Formulas. A few hundred years ago, formulas were written in prose. Today, mathematicians recognize the advantages of writing formulas in mathematical notation: they're shorter, easier to understand, and easier to manipulate. Replacing prose by mathematics must have reduced errors.

Mathematicians think they've stopped using prose to write formulas. They're wrong. They've replaced only some of the prose in their formulas by math. Consider this definition of what it means for $\lim_{x\to a} f(x)$ to equal b.

$$\begin{aligned}&\text{For all } \varepsilon > 0 \text{ there exists } \delta > 0 \text{ such that, for all } y,\\ &\text{if } 0 < |y-x| < \delta \text{ then } |b - f(y)| < \varepsilon.\end{aligned} \tag{1}$$

Leslie Lamport is a computer scientist at Microsoft Research.

A mathematician would find (1) perfectly normal, even though it's a mathematical formula written with many words. Here is that formula written without words:

$$\forall \varepsilon > 0 : \exists \delta > 0 : \forall y : (0 < |y-x| < \delta) \implies (|b-f(y)| < \varepsilon). \tag{2}$$

I believe most mathematicians would find (2) harder to understand and uglier than (1). I expect mathematicians a few hundred years ago would have found $0 < |y-x| < \delta$ hard to understand and ugly.

Why write (2) rather than (1)? For the same reason we don't write *0 is less than the absolute value of...*: It's shorter, easier to understand (when you become comfortable with the notation), and easier to manipulate. And it will reduce errors. Show elementary calculus students the definition (1) and ask them to write what it means for $\lim_{x\to a} f(x)$ equals b to be false. I doubt if many of them would get it right. Teach them a little elementary logic and they could easily compute the negation of (2). The most obvious use of words in formulas is to express logical operations; but they are also used in other ways, such as describing sets and functions.

Formulas written without words can now be manipulated by computer programs. Programs can easily compute the negation of (2). They can't compute the negation of (1).[1] Those programs can help students become comfortable with mathematical concepts, if the concepts are described with math rather than prose.

Mathematicians think it's difficult to write formulas completely mathematically, without words. I have asked a number of mathematicians how long a completely rigorous, wordless definition of the Riemann integral would be—assuming definitions of the set of real numbers and its arithmetic operations, as well as simple set theory. I've received answers ranging from 50 lines to 50 pages. They're wrong.

I've developed a language called TLA$^+$ that engineers use to write completely formal mathematical descriptions of computer systems. It has tools for checking the correctness of their mathematics. The Riemann integral can be defined in TLA$^+$ in about a dozen lines.

Proofs. A few hundred years ago, proofs were written in prose. They still are. Mathematicians haven't even begun to change the way they write proofs. They think their proofs express rigorous logical reasoning. They're wrong. Their prose proofs are written in a literary style that obscures the logic of the proof. Consider the following opening sentence of a proof from an elementary calculus book

[1]*Restricted, unnatural languages have been proposed for writing formulas approximately like* (1) *so they can be understood by a computer program. Such languages are of little or no use to people not afraid of mathematics.*

by Michael Spivak [20, page 170]—a book that is considered to be very rigorous.

> Let a and b be two points in the interval with $a < b$.

It is obviously wrong because the interval in question could consist of a single point, so it might be impossible to choose a and b. That sentence is actually part of a correct proof, but the reader must discover for herself the proof hidden inside Spivak's prose.

Writing proofs with prose leads to errors. How can those errors be avoided? Most mathematicians and computer scientists believe that the only way is to write machine-checked proofs. This requires writing formulas in a formal language. TLA$^+$ is simple enough that mathematically unsophisticated engineers can use it, and it is enough like everyday math that mathematicians should find it fairly natural. But it's too simple to be adequate for writing the kinds of proofs found in most math journal articles. Formalizing such proofs requires a language too complicated for most engineers to learn—one that I believe most mathematicians would find quite obscure. Few mathematicians would go to the effort of learning such a language unless it made writing their proofs significantly easier. Today, it makes writing most proofs much more difficult. Routine machine-checked proofs are now practical in just a few situations, including some safety-critical applications. I don't expect this to change in the next couple of decades.

Fortunately, there is a simple method that anyone can use now to write proofs with fewer errors. It can't eliminate all errors, but it can make them much less likely to occur. Its basic idea is to replace the linear order of ordinary prose by a hierarchical structure, and to name hypotheses and proved facts so they can be referred to later in the proof. Here is a brief explanation of the method; a complete description has appeared elsewhere [13].

A theorem consists of a statement together with its proof. A proof is either a short paragraph or a sequence of statements and their proofs. At each point in a proof, there is a current goal and a set of usable facts that can be assumed in proving that goal. Statements can be written in prose or in math. When written in math, the logical structure of the statement often determines the hierarchical decomposition of its proof. Figure 1 shows the structure of part of a proof containing the statement $A \wedge B \implies C$, in which C is proved by first proving statements D and E. Usually, those two statements would easily imply C, making the proof of qed step ⟨3⟩4 simple. The number ⟨2⟩3 indicates that it is the third statement in the level-2 proof of a level-1 statement.

This is a straightforward proof, and presented in this way there seems no reason to structure it. But suppose it were a small part of a large proof, and the proofs of D and E were each half a page long. If the proof were written as prose, how could the reader keep track of where the scope of the hypotheses A and B ended, and where it was no longer valid to use D? Mathematicians try to handle complexity by using lemmas; but that just provides one level of hierarchy, which doesn't get you very far.

Making a proof more rigorous requires filling in all the gaps that could conceal errors. This means making it longer. Making a prose proof longer makes it harder to read. But with hierarchical structure, the extra length makes the proof easier to read. The additional explanation appears at lower levels of the hierarchy, so it doesn't obscure the structure of the proof. This will be especially true when mathematicians stop producing pictures of print on dead trees and start using hypertext, so lower-levels of the proof can be hidden when not being read. Avoiding errors requires more detailed proofs than are currently found in journals. Until journals use hypertext, this means writing a detailed proof to catch errors, then shortening it for publication. That's easy to do with structured proofs: you just replace the lower levels of the hierarchy with short proof sketches. (One can even write LaTeX macros so a single file can produce either version by changing a few characters.)

Students can learn to write structured proofs by teaching them to write very simple machine-checked proofs in some field. Any good proof system should allow hierarchical structuring of proofs. The language for writing theorems should be simple—not the kind of complicated language needed for serious math. TLA$^+$ and its proof system

⟨2⟩3. $A \wedge B \implies C$

Current goal set to $A \wedge B \implies C$

  ⟨3⟩1. SUFFICES ASSUME A, B
        PROVE C

    Proof: By simple logic. Trivial proof that assuming A and B, then proving C, proves the current goal.

Current goal set to C; and A and B added to usable facts.

  ⟨3⟩2. D

    Proof of D

D added to usable facts.

  ⟨3⟩3. E

    Proof of E

E added to usable facts.

  ⟨3⟩4. QED

    Proof of C

Current goal and usable facts same as before ⟨2⟩3 except with fact $A \wedge B \implies C$ added.

⟨2⟩4. ...

Figure 1. A statement and its structured proof.

are not ideal, but they could be used if nothing better is available.

Students should understand that the facts they learn in their math classes can, in principle, be formally proved from simple axioms and proof rules. In practice, we only carry proofs down to the level where we believe the reader will find the steps to be obviously true. That level rises with education and experience. We also sometimes take shortcuts by writing formulas with words. But students and mathematicians should have the confidence that they could make their proofs completely rigorous and carry them down as close as they want to basic axioms.

Writing hierarchically structured proofs can help you avoid errors; it can't guarantee that you will. You have to be honest with yourself about what's obvious and what should be proved. My advice is to write the proof down to a level at which you think everything is obvious, and then go one level deeper. But if you don't care whether your proofs are correct, nothing short of having to write a machine-checked proof will keep you from making errors.

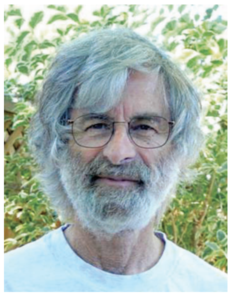
What should you do now? If you agree that writing formulas with words or that writing proofs with prose is silly, just stop doing it. You don't have to wait for others to change.

Formulas. You needn't remove all the words from your formulas. Start by using the quantifiers $\forall$ and $\exists$. Then try eliminating "…", which is not a mathematical operator. The sequence $x_1, \dots, x_n$ is just a function x with domain $1..n$ that maps each i in its domain to x_i.[2] Often, though not always, the math becomes simpler and more elegant if you eliminate the "…" and instead use the function x. Give it a try. Be aware of when you're using words and sloppy notation instead of being rigorous. If you're open to change, you will find that the mathematically rigorous approach is often the simplest. If you're a teacher, your students should have learned, or should be learning, the basic math needed to write formulas with fewer words than they now use. Help them to become more comfortable with proper mathematical notation by using it in your classes.

Proofs. There is no reason not to start writing structured proofs now. It takes only a sentence or two to explain to readers how to read them. I've been doing it for about 30 years, and no editor or referee has complained about my proofs. Start by reading how I write structured proofs, but feel free to modify my style as you see fit. There are just two features that should be preserved: hierarchical structuring and the ability to name and refer to hypotheses and already proved statements.

[2] *".." is a mathematical operator, defined by* $i..j \triangleq \{k \in \mathbb{Z} : i \leq k \leq j\}$, *where* $\mathbb{Z}$ *is the set of all integers.*

If you're a professor, teach your students to structure their proofs the way you do. They're not yet set in their ways, and they'll appreciate how the structure makes your proofs easier to understand. Encourage them to write structured proofs in all their courses. Other professors are unlikely to complain that the proofs are too rigorous; and they might even be inspired to write them themselves.


Leslie Lamport

5. What is a Proof? What Should It Be?

Christoph Benzmüller

Does the notion of a mathematical proof refer to the rigorous but typically rather nonintuitive *formal derivation of a new "truth" from its premises using accurately defined rules of inference*? Or is it an *artful communication act* in which the beautiful structures underlying a new mathematical insight are revealed to peers in such a way that they can easily *see* and accept it, and even gain further inspiration?

The former notion of a *formal proof* is primarily concerned with logical rigor and soundness. Intuition and beauty is a secondary concern, if at all. Formal proofs have recently attained increased, albeit quite controversial, attention in mathematics, triggered, e.g., by successful applications of modern automated theorem proving technology to challenging mathematical verification and reasoning tasks. Using theorem provers, hard problems were solved that humans could not manage on their own. Some examples include:

(i) *The four-color theorem:* this notorious challenge was solved already in 1977 using automated theorem technology by Appel & Haken, and an interactive formal

Christoph Benzmüller is the Chair for AI Systems Engineering and a professor at the Otto-Friedrich-Universitä, Bamberg; he is also a professor in the Department of Maths and Computer Science at the Freie Universität, Berlin. His email address is `c.benzmueller@gmail.com`.

proof was more recently developed by Gonthier in the proof assistant Coq.

(ii) *The Kepler conjecture* (about best sphere packings in Euclidean 3-space): a board of experts had gone to great lengths to verify the solution to the problem submitted by Hales to the *Annals of Mathematics*, but in the end could not fully verify his contribution. Eventually, Hales and his team mastered it in interaction with HOL Light. As the main result, a formal proof is now available that is independently verifiable—by humans and computer programs.

(iii) *The Pythagorean triple problem* (whether all positive numbers can be colored by two colors so that there is no monochromatic Pythagorean triple): this challenge problem was solved by Heule, Kullmann, and Marek [9] using automated SAT solving technology (boolean SATisfiability), and the formal proof that was generated by the computer program is of enormous size (about 200TB); it is still independently verifiable though (at least by machines). This line of research has been continued by an automatic solutions for Schur number five [8] and Keller's conjecture [5].

(iv) My own current work with colleagues focuses on higher-order metalogical reasoning technologies [3] that have enabled us to detect and explain errors and problems in peer-reviewed publications in mathematics, computational metaphysics, and machine ethics. These include the discovery of an unnoticed inconsistency in Gödel's modern variant of the ontological argument for the existence of God, the discovery and clarification of a deeply rooted paradox in Zalta's *Principia Logico-Metaphysica*, and the revelation of some minor problems in a well-known textbook on category theory [4].

There are many other works that could be mentioned here, e.g., the machine-checked proof on the odd order theorem.

It is to be noted that the formal reconstruction of mathematical work is generally a very resource- and time-consuming task, for example due to the lack of large and easily reusable libraries of formalized mathematics, or software that does not pose additional challenges along the way. Conversely, some technical mathematical results, such as those mentioned in (iii), may not support intuitive and insightful mathematical proofs. In general, mathematics is confronted with increasingly complex problems whose solution and subsequent evaluation (e.g., by peer review) require techniques that go beyond traditional practice. Examples (i)–(iv) above are only early evidence of this kind. When technologies are available that can help detect errors in publications, they should certainly be used to optimize scientific quality. Formal proofs should therefore take a more central role, in maths and beyond.

In fact, I see it as a societal duty to take up this challenge. Clinging to the traditional notion of mathematical proof alone is not an option in an increasingly technological world. Think of areas such as "program verification in computer science" or "trusted AI," where ideally we want formal guarantees that implemented, complex solutions are mathematically correct, but where intuitive, traditional proofs might be lacking.

Nonetheless, formal proofs alone are of limited interest and they should ideally always be coupled to intuitive proofs. Explainability, transparency, and intuition must remain virtues of the highest priority, not only in mathematics. In the long run, the increased trustworthiness and beauty of a combined approach, where both notions are coupled pari passu, will justify the additional resources that must now be invested. Publishing errors (even minor ones) will be prevented by formal proofs, and designing ill-conceived and inaccessible theories will be impeded by the demand for mathematical intuition and beauty.

So, what should a proof be? In conclusion, it should ideally be both a human-oriented traditional proof and a machine-oriented formal proof. Traditional mathematical proofs are made by, and consumed by, humans, while formal proofs are predominantly generated with, and consumed by, machines. Yet, by pairing both, the antipodes will increasingly engage in a harmonious dance.

One step further, beyond *pairings*, one may dream of the *integration* of mathematical and formal proofs into one object. Modern proof assistants such as Isabelle/HOL provide increasingly intuitive languages for constructing and representing proofs, but significant scientific progress is still needed to achieve this ambitious goal. Finding a comprehensive solution that integrates both notions of proof can even be understood as a challenge for AI, as it requires a seamless semantic integration of natural language, diagrams, formula language, etc., up to the exchange of meaningful arguments between humans and theorem provers.

Recent work by Marco David, Benedikt Stock, Jonas Bayer, and their fellow students gives reason for hope. Their verification project of Hilbert's tenth problem is significantly smaller in scope than some projects mentioned above, but differs in that the mathematics students began their formalization project, encouraged by Matiyasevich, with no prior knowledge of proof assistant technology. Nevertheless, they rose to the challenge and made great progress by working independently with the proof assistant. This is particularly notable because it is another example (in addition to, e.g., [6]) of an impressive formalization project carried out by people from outside the community of formalized mathematics, and it is another excellent demonstration of the maturity that modern proof assistant technology has now reached.

Christoph Benzmüller

6. Proof Assistants: The Right Way to Learn Mathematics?!

Jonas Bayer, Marco David, and Benedikt Stock

Little did we know at the time that Yuri had made a "grand plan" to establish his vision on the future of mathematical proof when we first met him. He visited our university back in 2017 and presented the idea to formalize the DPRM theorem:[3] this is the key result in his proof of the undecidability of Diophantine equations, providing a negative solution to Hilbert's tenth problem. Little did we know that we played the role of "guinea pigs" who, in a carefully set-up experiment, should demonstrate the idea that mathematical proof verification by computers is feasible even for young and inexperienced university students. Today, we are very grateful for the role we have been given. The following paragraphs elaborate what we have learned in the years since then, now that we are ourselves supervising another project by a fresh set of students that had never worked on computer-verified proofs before.

Yuri set up the scene swiftly; he presented his theorem to us and invited us to work on its computer verification. Neither he nor we had any idea how much effort this would turn out to be. At the time, it seemed to us that, once the proof of the DPRM theorem was understood, we merely needed to "translate" its arguments in such a way that Isabelle could verify them. Anyone who has ever worked with an interactive theorem prover, however, knows that the word "translate" does not do much justice to the process. In reality, this includes filling the possible gaps that are frequently found in mathematical papers. Common phrases such as "it is easy to see that..." needed to be brought to logical life during this process. Soon, we realized that the challenges often lay in lemmas that looked rather innocent. In fact, we found ourselves five times reconsidering the formalization of the mere concept of a register machine before our formal definition proved useful.

Despite the challenges we encountered, we insist that learning how to work with an interactive theorem prover is fully feasible, although not yet easy. In a previous paper [2], we have reflected upon our own mistakes and the challenges we encountered. A key realization is that making progress is almost impossible without an expert at hand to answer questions. Hence, to popularize proof assistants among the next generation, we are now proactive ourselves in mentoring a new group of students learning how to formalize mathematics. The current need to pass on experience person-to-person distinguishes proof assistants from most programming languages or computer algebra systems like Mathematica. Here, online Q&A forums such as Stack Overflow[4] provide a freely accessible and searchable database of almost any imaginable question about these tools, and hence provide the means to make them accessible to the broad public. Interactive theorem provers and their currently secluded communities must follow suit to become useful for the average mathematician!

Beyond the technical skills and results, this project prompted a paradigm shift in the way we now view ordinary mathematics: In our university courses, we no longer ask if a proof is convincing to us, but instead wonder if we would be able to formalize it in Isabelle. As the computer often exhibits assumptions or edge cases that humans gloss over, this thought pattern leads to a more rigorous approach to the (informal) argument. Our new, sharpened perspective was largely forged through the interaction with the computer and its unique manner of reasoning. It illustrates how to reconcile the two conflicting definitions of "proof" in the next generation of mathematicians.

Jonas Bayer is a PhD student at Cambridge University. His email address is `jcb234@cam.ac.uk`.

Marco David is a PhD student at the University of California, Berkeley. His email address is `marco.david@berkeley.edu`.

Benedikt Stock is a PhD student at the University of Oxford. His email address is `stock@maths.ox.ac.uk`.

[3]*Named after Martin Davis, Hilary Putnam, Julia Robinson, and Yuri Matiyasevich, the DPRM theorem states that every recursively enumerable set is Diophantine.*

[4]*https://stackoverflow.com*

Jonas Bayer

Marco David

Benedikt Stock

7. Some Thoughts on the Formalization of Mathematics

Kevin Buzzard

Making formalization sexy. Mathematics has fashions. The Langlands philosophy seems to have been a fashionable area for many decades now. Conjectures get proved, new conjectures get made, there was the classical theory, and then a mod p theory (crucial for Fermat's Last Theorem), and now a bewildering p-adic theory as well as geometric and categorified versions. I will unashamedly confess that my work with Johan Commelin and Patrick Massot where we translated the line "Let X be a perfectoid space" into Lean's language was an attempt to *market* theorem proving software to mathematicians. We wanted to show them that theorem provers are now ready to handle fashionable modern mathematics. No, we cannot do anything spectacular yet like come up with an incomprehensible billion-line long proof of the Riemann hypothesis (indeed, computers generating proofs of hard mathematical conjectures of mainstream interest is science fiction right now, and may well remain so for decades to come). However, let's think about this. If we stick with the status quo (i.e., essentially nobody types sexy mathematics into proof assistants), then proof assistants will *never* be able to do sexy mathematics, because they will simply never learn it! Computers certainly can't directly read the crap we write in our papers (and humans often can't read them either). Humans need to do the translation by hand, and the sooner the better. So, whose *job* is it to type in the *statements* of the global Langlands conjectures into these systems? Who will do this, and give humanity the chance to make computer teaching and proof mining resources, and large language model training data, for students learning and working in the Langlands program? Surely it is our job as mathematicians. Nobody else is going to do it – we cannot expect computer scientists to become technical experts in the Langlands philosophy; it is much easier for mathematicians to learn a programming language. If the maths staff won't do the translation, let's get the maths PhD students doing it. If you've recently graduated with a PhD from a pure mathematics department, can you, or even can humanity, *state* in Lean the main theorems you proved in your thesis? The practical answers with current technology are still highly dependent on the nature of the area of the thesis. For my MSc and PhD students I am 100% certain that we can not only state the main results in Lean, but also prove them.

Looking to the future, what if mathematicians start flocking to formalization, and all of a sudden we have got perfectoid spaces and the statements of the Langlands conjectures in Lean and also in Arend (a HoTT prover) and Isabelle/HOL (a simple type theory prover) and MetaMath (a set theory prover) and Coq and HOL 4 and HOL Light and Mizar and cubical Agda and.... What then? I have already said that, in my opinion, it is science fiction to expect that these systems will start *proving* the Langlands philosophy.

But the following is near-reality. Software such as Lean can be used to power an interactive resource for PhD students learning algebraic geometry or some other dense area. Once we have a database of *statements* of many theorems corresponding to tags in the Stacks Project or Kerodon (online databases of algebraic geometry and other sexy mathematics), we can let the computer scientists take over. They have tools known as hammers, which can attempt to build proofs or counterexamples to propositions fed to the system, using the database we mathematicians constructed. Note that such a tool *does not need any formalized proofs* and so even a huge database like the the statements of the theorems in the Stacks project could be constructed manually over a period of several person-years, ideally by young algebraic geometers interested in trying new ways of learning the area. Of course, formalising the proofs is the fun part, so these will no doubt follow.

Kevin Buzzard is a professor of pure mathematics at the Imperial College of London. His email address is `k.buzzard@imperial.ac.uk`.

Another possibility is training large language models such as ChatGPT to write Lean code. This area is still in its infancy but may turn out to be decisive.

Making formalization fun. The examples in the previous paragraph show that the mathematical community might well come to benefit from having serious mathematics formalized in a theorem prover. Experience suggests that it is mostly young people who formalize. Thus teaching young people about formalization is important; however it is a different task to teaching young people about mathematics. The Natural Number Game is a browser-based Lean game that came out of many hours at Imperial College London just writing random undergraduate-level mathematics puzzles in Lean and then watching students solving them. The idea of building up facts about natural numbers from first principles was a big hit, and it went on from there. Students say to me "I've finished the natural number game, what next?" The correct answer to that is "Install Lean following the instructions on the community website". But after they've done that, what next? This depends on their mathematical interests and abilities. I often encourage an enthusiastic student to formalize some mathematics they already know well for their first Lean project. The rule is: if it compiles, you won. After this we can start talking about how to write code which is of a high standard.

For PhD students and more mathematically mature people who are interested in seeing what's going on, it's always worthwhile pointing out current mathematical projects which are run via Zulip at `leanprover.zulipchat.com`, the platform behind the Lean community chat. We are always looking for new people to help out with various projects, and are happy to "onboard" newcomers. Right now some of the main active projects on the site are: a formalization of the proof of Fermat's Last Theorem for regular primes, being led by Riccardo Brasca, and a formalization of some of the main results in the theory of condensed mathematics, being led by Adam Topaz. A future project, starting in 2024, led by me and funded by the EPSRC, is to formalise a full proof of Fermat's Last Theorem. Past projects which ran on the site include the sphere eversion project, led by Patrick Massot, which formalised a modern proof that one can evert a sphere in three dimensions, and "the liquid tensor experiment", led by Johan Commelin, formalising the proof of a 2019 theorem of Clausen and Scholze. In 2020 Scholze challenged the formalization community to verify a crucial theorem he had announced with Clausen, and the Lean community rose to the challenge. The back story is interesting; Scholze suggested that perhaps the current refereeing process that we have in mathematics would not dive deeply into a specific part of the work, and Scholze was interested to see whether a theorem prover could do this instead. It turns out that this was indeed possible. Scholze was an advisor throughout the project, which took 18 months to complete; on the way the community developed many other theories (for example a theory of homological algebra), some of which are now being used in other projects.

Teaching formalization skills. Teaching students how to formalize mathematics means teaching them how to translate mathematical ideas between English and Lean. Just like learning a foreign human language, you begin by translating basic stuff between the two languages, and you ask if you don't understand something. In the Lean course I teach in Imperial College London's mathematics department, we work through human mathematics that the students have already been taught, and we learn to speak Lean's language. Later on I introduce some new mathematics, when the students have already learnt some Lean. Athina Thoma and Paola Iannone taught me that teaching the first years equivalence relations and Lean at the same time would usually not end well. However to a student who knows both, formalising the fact that equivalence relations biject with partitions would be an excellent Lean learning experience. The art is to find the right project, and the correct project typically depends on the student.

Fixing issues in modern research. In an interview with *Wired* I was once quoted as suggesting that all maths was wrong. This is something I know not to be true—some maths is definitely correct, and (at least in classical logic) these statements are opposite to one another. However, I did say that perhaps some of our castles were built on sand, and I now think even this is a bit naive. Having had conversations with David Rabouin, a historian and philosopher of mathematics, I now understand that cutting edge maths always looks like this. There are bits which are not quite checked but everyone knows it will usually work out fine, or at least well enough to make the main theorem go through.

Sometimes maths does go wrong though. This century, in my area alone (number theory), prestigious mathematicians have announced proofs of Leopoldt's conjecture and the ABC conjecture; the latter work was even published in a reputable mathematical journal. Yet our community does not seem to accept the proofs as rigorous. Unfortunately, Lean will not deliver us from these problems, at least not yet. Right now, Lean proves theorems by having humans translate those theorems from the English, and if nobody is prepared to take on the monumental task of translating the Mochuzuki proof of the ABC conjecture into a theorem prover (and why should anybody? This is not how the mathematical community has treated published proofs in the past), then I don't see any way past the impasse.

Some proofs have already gone beyond the one brain barrier—no one human understands all of the ideas in it. Areas which were formerly fashionable can die out if the big conjectures driving them forwards are resolved. Anything not properly documented actually runs the risk of being lost. One can hope that new and simpler proofs will come along. But history shows that this is not always true: sometimes things are complicated, and stay complicated. Whether or not we choose to use theorem provers to do it, mathematicians need to start thinking more carefully than ever about precisely documenting what we think we already know, so we can answer technical questions from future generations of mathematicians.

Kevin Buzzard

8. When Will Computer-Assisted Proof Become Part of Everyday Mathematics?

Lawrence Paulson

Introduction and background. The idea of applying technology to mathematical reasoning began to be realized in the 1960s. N. G. de Bruijn's AUTOMATH was a type theory for expressing mathematical definitions and proofs. Trybulec's Mizar system included a human-readable formal language for abstract mathematics. Both were professional mathematicians and intended their work to be beneficial to mathematics itself, but the technology was not ready. Nevertheless, they made valuable progress. AUTOMATH led to the dominant type theory today, the Calculus of Inductive Constructions. Mizar accumulated a substantial and wide-ranging library of formalized mathematics, while its readable structured language is still the best there is.

Interactive theorem provers, or proof assistants, emerged in the 1970s. The earliest is arguably Mizar, but the most influential was undoubtedly Edinburgh LCF. It implemented a Logic for Computable Functions that promptly became obsolete, but it also introduced an architecture that would be adopted by most successor systems, including Coq, HOL, Isabelle, and Nuprl [7]. These tools were intended for verification in computer science. HOL (for higher-order logic) had been chosen in order to implement a certain style of hardware verification. These verification tasks seldom required any mathematics beyond the integers.

The 1994 discovery of a bug in the floating-point unit of Intel's Pentium dramatically focused the verification community on the real numbers. John Harrison formally proved the correctness of an algorithm for computing the exponential function, taking account of all the peculiarities of floating-point arithmetic. He went on to play a major role in the *Flyspeck project*: the formal confirmation of Thomas Hales's proof of the Kepler conjecture. He formalized many landmark results in mathematics, such as the prime number theorem.

Gonthier's formalization of the four-color theorem had already demonstrated that an interactive theorem prover (Coq in this case) could help settle a genuine question in mathematics. This task was similar to Flyspeck in that it involved formally checking a large number of computations.

Isabelle. Isabelle [18] emerged from the LCF tradition with the aim of supporting a multiplicity of formalisms, including set theory. However, with higher-order logic dominating the verification world, Isabelle/HOL became the dominant instance of Isabelle. Its formalism extended that of the various HOLs with axiomatic type classes, allowing the systematic reuse of formal material sharing the same axiomatic basis [10]. Isabelle adopted a structured proof language, Isar, based on Mizar's mathematical language. Isar expresses proofs with a nested structure where milestones—intermediate claims and proofs—are explicitly written out. Isabelle provides automation for proof (powerful external provers can be invoked through *sledgehammer*) and also for disproof in the form of *counterexample search*. The user interface is a unique interactive development environment for editing live proof documents.

There are powerful synergies between these features. Structured proof text is easy to reuse. Copied into a new development, it will instantly be checked and any errors flagged. Structured proofs also work well with sledgehammer: if a given statement is too difficult to be proved automatically, the user may propose an intermediate statement that might be easy enough to prove automatically and lead eventually to a proof of the original statement.

This powerful automation has greatly relieved the tedium that accompanied formal proof in the early days.

Lawrence Paulson is a professor of computational logic at the University of Cambridge. His email address is `lp15@cam.ac.uk`.

It has also helped encourage the growth of substantial libraries of formalized mathematics.

Proof assistants: Ready for prime time? Today we have a wide variety of systems: the HOL family, for higher-order logic; Isabelle/HOL, for higher-order logic with axiomatic type classes; Coq and Lean, implementing the Calculus of Constructions. These choices are motivated by conflicting priorities such as the simplicity of the implementation and its semantics as opposed to the expressiveness of the logical calculus. Competition between research groups was spurred by Freek Wiedijk's online list detailing which of the "top 100 theorems" [23] had been proved in various systems. Only one of the hundred theorems remains unproved, and that is Fermat's Last Theorem!

Much of modern mathematics falls within the scope of existing verification tools. There are now huge libraries of formalized mathematics. Isabelle's *Archive of Formal Proofs*[5] contains over 650 entries including much core mathematics—linear algebra, multivariate analysis, probability, complex analysis, topology—and over three million lines of proofs. Lean's *mathlib* is an immense and rapidly-growing corpus of material from every branch of mathematics.[6]

So are these systems finally ready to support mathematicians? There are still many obstacles:

- Formal syntax looks artificial and often is barely legible. To see the difficulties presented by traditional notation, contrast x^2, $\nabla^2 f$, $\sin^2\theta$, $f^2(x)$. In group theory, G denotes a group but also a set; ab is the product of a and b but HK is quite a different thing, simply because we used a different part of the alphabet. In set theory, $\lambda < \aleph_1$ has a different meaning from $\lambda < \omega_1$ even though $\aleph_1 = \omega_1$.
- The libraries of formalized mathematics still have many gaps, and what they do have is difficult to find. The names of theorems are frequently ambiguous (what is Roth's theorem?), as are concepts such as limits (which could refer to analysis, topology or even set theory). But one should be able to search for "limit theorems" and get something relevant. Better still, the system might make suggestions unprompted.
- Obvious statements are often too hard to prove. Examples include showing a set to be finite, showing a function to be continuous, calculating a derivative and evaluating a limit. In some cases, skilful use of the existing automation can be effective. In others, specialized decision procedures must be coded. Best known are decision procedures for linear arithmetic; a recent development is Manuel Eberl's limit solver.

Toward the future. Recently, with funding from the European Research Council,[7] my colleagues and I have been exploring and stretching the limits of today's technology. We have formalized relatively recent (post 1970) and deep results.[8] The latter include additive combinatorics, extremal graph theory and combinatorial design theory. We have even formalized some sophisticated definitions, notably Grothendieck schemes—necessary for advanced work in algebraic geometry—which had hitherto been thought to lie outside the scope of Isabelle/HOL's simple type theory.

Mathematicians also need help navigating our huge library of formalized mathematics. It can be hard to know whether a desired result has been formalized: many theorems go by various names, or conversely, one name (e.g., Young's inequality) may be applied to a family of distinct results. For every well-known result, there may be dozens of technical lemmas, mostly obvious and yet needed in order to prove anything on today's systems. My colleagues have been building an experimental search engine, called SErAPIS,[9] that makes it possible to search the entire library with the help of a huge dictionary of mathematical concepts.

Also attractive is the idea that the proofs in our library might be used to generate new proofs automatically. This is another way the computer can help the user get value out of our three million lines of formal proofs. Promising results are just starting to appear from a number of research groups.

For the further future, we may hope to mechanize mathematical intuition. This is the knowledge that tells us that a given function is surely continuous or that a certain formula cannot generate only prime numbers. That's how we know that a particular claim cannot be true as stated or proved using the methods advertised. Formalization has demonstrated time and again that while published proofs often contain errors, theorem statements are generally correct. Mathematicians can perceive the truth if they can't always write down the correct argument. Giving such intuition to a computer would transform our field. This task will remain open for the next generation or two.

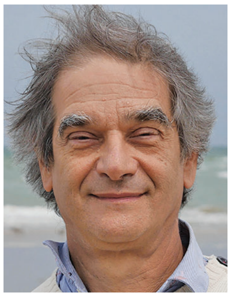
[5] *http://www.isa-afp.org*

[6] *https://leanprover-community.github.io/mathlib-overview.html*

[7] *Project ALEXANDRIA, GA 742178*

[8] *https://www.cl.cam.ac.uk/~lp15/Grants/Alexandria/*

[9] *https://behemoth.cl.cam.ac.uk/search/*


Lawrence Paulson

References

[1] I. N. Baker, *Completely invariant domains of entire functions*, Mathematical Essays Dedicated to A. J. Macintyre, Ohio Univ. Press, Athens, OH, 1970, pp. 33–35. MR271344

[2] J. Bayer, M. David, A. Pal, and B. Stock, *Beginners' quest to formalize mathematics: A feasibility study in Isabelle*, in C. Kaliszyk, E. Brady, A. Kohlhase, and C. Sacerdoti Coen, editors, *Intelligent Computer Mathematics*, pages 16–27, Springer International Publishing, Cham, 2019.

[3] C. Benzmüller, *Universal (meta-)logical reasoning: Recent successes*, Science of Computer Programming **172** (2019), 48–62. DOI 10.1016/j.scico.2018.10.008.

[4] C. Benzmüller and D. S. Scott, *Automating free logic in HOL, with an experimental application in category theory*, J. Automat. Reason. **64** (2020), no. 1, 53–72, DOI 10.1007/s10817-018-09507-7. MR4058124

[5] J. Brakensiek, M. Heule, J. Mackey, and D. Narváez, *The resolution of Keller's conjecture*, J. Automat. Reason. **66** (2022), no. 3, 277–300, DOI 10.1007/s10817-022-09623-5. MR4449705

[6] K. Buzzard, J. Commelin, and P. Massot, *Formalising perfectoid spaces*, in J. Blanchette and C. Hritcu, editors, *CPP 2020*, pages 299–312, ACM, 2020.

[7] J. Harrison, J. Urban, and F. Wiedijk, *History of interactive theorem proving*, Computational logic, Handb. Hist. Log., vol. 9, Elsevier/North-Holland, Amsterdam, 2014, pp. 135–214, DOI 10.1016/B978-0-444-51624-4.50004-6. MR3362157

[8] M. J. H. Heule, *Schur number five*, in *AAAI*, pages 6598–6606, AAAI Press, 2018.

[9] M. J. H. Heule, O. Kullmann, and V. W. Marek, *Solving and verifying the Boolean Pythagorean triples problem via cube-and-conquer*, Theory and applications of satisfiability testing—SAT 2016, Lecture Notes in Comput. Sci., vol. 9710, Springer, [Cham], 2016, pp. 228–245, DOI 10.1007/978-3-319-40970-2_15. MR3534782

[10] J. Hölzl, F. Immler, and B. Huffman, *Type classes and filters for mathematical analysis in Isabelle/HOL*, Interactive theorem proving, Lecture Notes in Comput. Sci., vol. 7998, Springer, Heidelberg, 2013, pp. 279–294, DOI 10.1007/978-3-642-39634-2_21. MR3111278

[11] IMU Committee on Electronic Information and Communication (CEIC), World digital mathematics library (WDML). `https://www.mathunion.org/ceic/library/world-digital-mathematics-library-wdml`.

[12] L. Lamport, Errors in proofs—a correction and further data. `https://lamport.azurewebsites.net/tla/proof-statistics.html`.

[13] L. Lamport, *How to write a 21st century proof*, J. Fixed Point Theory Appl. **11** (2012), no. 1, 43–63, DOI 10.1007/s11784-012-0071-6. MR2955006

[14] D. Larchey-Wendling and Y. Forster, *Hilbert's Tenth Problem in Coq* (en), Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik GmbH, Wadern/Saarbruecken, Germany, 2019, DOI 10.4230/LIPICS.FSCD.2019.27, `http://drops.dagstuhl.de/opus/volltexte/2019/10534`.

[15] Yu. V. Matiyasevich, Математическое доказательство: вчера, сегодня, завтра (Mathematical proof: yesterday, today, tomorrow), in Компьютерные инструменты в образовании (Computer instruments in education) **6** (2012), 13–24. `http://ipo.spb.ru/journal/index.php?article/1532/`.

[16] K. Pak, *Diophantine sets. Preliminaries*, Formalized Mathematics **26** (2018), no. 1, 81–90, `http://dx.doi.org/10.2478/forma-2018-0007`, DOI 10.2478/forma-2018-0007.

[17] K. Pak, *The Matiyasevich Theorem. Preliminaries*, Formalized Mathematics **25** (2017), no. 4, 315–322, `http://dx.doi.org/10.1515/forma-2017-0029`, DOI 10.1515/forma-2017-0029.

[18] L. C. Paulson, *Isabelle: A generic theorem prover*, Lecture Notes in Computer Science, vol. 828, Springer-Verlag, Berlin, 1994. With contributions by Tobias Nipkow, DOI 10.1007/BFb0030541. MR1313213

[19] L. Rempe-Gillen and D. Sixsmith, *On connected preimages of simply-connected domains under entire functions*, Geom. Funct. Anal. **29** (2019), no. 5, 1579–1615, DOI 10.1007/s00039-019-00488-2. MR4025520

[20] M. Spivak, *Calculus*, W. A. Benjamin, Inc., New York, 1967.

[21] B. Stock, A. Pal, M. A. Oprea, Y. Liu, M. S. Hassler, S. Dubischar, P. Devkota, Y. Deng, M. David, B. Ciurezu, J. Bayer, and D. Aryal, *Hilbert Meets Isabelle: Formalisation of the DPRM Theorem in Isabelle*, EasyChair, 2018, DOI 10.29007/3q4s, `http://dx.doi.org/10.29007/3q4s`.

[22] W. P. Thurston, *On proof and progress in mathematics*, Bull. Amer. Math. Soc. (N.S.) **30** (1994), no. 2, 161–177, DOI 10.1090/S0273-0979-1994-00502-6. MR1249357

[23] F. Wiedijk, Formalizing 100 theorems. `http://www.cs.ru.nl/~freek/100/`.

Credits

Photo of Christoph Benzmüller is courtesy of Pressestelle Erzbistum Bamberg / Dominik Schreiner.

Photo of Kevin Buzzard is courtesy of Imperial College London/Thomas Angus.

All other author photos are courtesy of the respective authors.