Why formalize mathematics?

Patrick Massot*

December 5, 2021

Abstract

We've been doing mathematics for more than two thousand years with remarkable success. Hence it is natural to be puzzled by people investing a lot of time and energy into a very new and weird way of doing mathematics: the formalized way where human beings explain mathematical definitions and proofs to computers. Beyond puzzlement, some people are wary. They think the traditional way may disappear, or maybe even mathematicians may disappear, being replaced by AI agents. These events are extremely unlikely and they are not the goals of the mathematical formalization community. We want to *add to our tool set*, without loosing anything we already have. In this text I'll explain what we want to add, distinguishing what already partially exists and what is currently science fiction. Examples will use Lean, a proof assistant software developed mostly by Leonardo de Moura at Microsoft Research, but everything I'll write applies to other proof assistants such as Coq or Isabelle.

1 Checking

The most obvious benefit of formalized mathematics is super-human certainty that a proof is correct when it has been checked by a computer. The precise meaning of this is complicated and it's not clear at all how this level can be reached while so many bugs exist in software. There are very good answers to these questions, but they are not the topic of this text. I also won't explain why we want proofs at all since this isn't specific to formalized mathematics.

One especially relevant aspect is that a super correct proof ensures completeness and consistency at all scales. Some small scale consistency is rather easy to fix anyway, although it's good to get help from the computer to make sure there is no forgotten corner case (maybe involving the empty set or the only even prime number). Trickier completeness involves making sure there are no wrong implicit claims. At medium scale, consistency means we won't tweak a definition to make sure Lemma 12 works and fail to notice that Lemma 1 is now broken. Large scale consistency means that we don't allow misunderstanding to creep in by using a theorem from a paper that had a slightly different standing assumption or notation.

Computer guaranteed completeness is not only reassuring for mathematicians, it is the ultimate dream of referees. It allows to focus on importance and novelty of ideas while evaluating a paper. But the current technology doesn't allow to hope we will soon ask for a formal proof with any submitted paper. Formalization is simply too time-consuming

*Université Paris-Saclay, CNRS, Laboratoire de mathématiques d'Orsay, 91405, Orsay, France.

for now, and we don't have enough formalized basic mathematics to build upon. So we currently pursue more focused goals.

First we can concentrate on rather small very technical parts of a larger story. This is what happened with the liquid tensor experiment [Sch20]. This challenge sits in the huge context of condensed mathematics by Clausen and Scholze aiming to build a better framework for endeavors mixing arithmetic, geometry and functional analysis. But the argument that needed careful checking only required a small portion of this context. In that case we had enough preexisting work to run a complete formalization starting from the axioms [Sch21]. But nothing prevents adding well known results as unchecked building blocks. The software will notice and report this, but it can still keep going. The correctness guarantee is correspondingly much lower because errors can hide in the unproven statements, but it can still be higher than in traditional mathematics.

Another possible focus is to check a proof that is simply too big for human brains. One way this can happen is when a *lot* of computation is involved, for instance in the four colors theorem [Gon07] or the Kepler conjecture proof [Hal+17]. Those two proofs were originally using unchecked computer assisted calculations, but have since been formalized. This kind of proof represent an absolutely tiny fraction of mathematics, but it seems this fraction is slowly growing so it's good to have tools to do this properly.

Other big proofs involve no large computation at all. The most famous example is the classification of finite simple groups. The size of this proof is so huge that nobody can guarantee its large scale consistency and its status is accordingly somewhat controversial. More modern examples include work in the Langlands program that use results from so many different branches of mathematics than nobody can master all of them, or foundational aspects of symplectic topology that are extremely difficult to get right. Here formalized mathematics could help, but this is science fiction up to now. We would need much more efficient tools to work at this scale in reasonable time.

2 Explaining and learning

Checking correctness is important and is the area where formalized mathematics has indisputable superiority over the traditional ways. But I don't think this is the most compelling reason to formalize mathematics.

Mathematicians spend a lot of time writing and reading mathematical explanations. In the traditional process, the writer needs to choose the assumed background knowledge and select a level of detail. Assuming background knowledge is necessary because we can't redo everything from the axioms each time we want to explain new pieces of mathematics. Choosing a detail level is needed because explaining all details would be extremely difficult and, more importantly, the result would be unreadable. Those choices are tricky and being consistent is difficult. The situation is anyway hopeless unless there is exactly one intended reader and reading time. Then readers have almost no choice. The only possible action influencing those choices is to skip portions of the text.

Having better tools to explain mathematics is a very important goal not only because people want to learn elementary mathematics or shiny new research level mathematics. It can also prevent mathematical understanding to get lost forever. For instance, people currently fear that some very important and deep parts of geometric topology could get lost forever, for instance the Kirby-Siebenmann theory or Freedman's classification of simply connected 4-manifolds. Specialists of this topic are getting old, young people don't learn the details and, more importantly, the original papers were written for specialists

fully immersed in the theory (there is popular science account of the case of Freedman's classification in [Har21]).

I think the most the promising application of formalized mathematics is the dream of producing mathematical documents allowing *readers* to dynamically choose the detail level and access background knowledge on demand. This requires the extreme precision of formalized mathematics to give access to the extremely detailed end of the detail range. Formalized mathematics could also indirectly contribute to the other end of the detail range by freeing resources for very informal accounts. And it could be psychologically easier for authors to write very heuristic explanations when having their back covered by the formal proof.

The background knowledge freedom is ensured by the fact that formalized mathematics can start from bare axioms in a fully reusable way. A key point is that the computer has super-human patience to repeatedly check existing explanations or tiny variations thereof. When working on an advanced explanation, a modification of a basic explanation can break its preexisting applications. But such an accident is immediately detected by so-called continuous integration techniques which check overall consistency at every single change.

The notion of detail level is pretty subtle. One immediate interpretation is that it deals with the amount of reasoning steps. A more detailed explanation can include more tedious checks that all assumptions of all lemmas are satisfied (for instance these could be checking continuity of some function or commutativity of certain natural diagrams). Allowing to choose detail level in this sense requires a structured proof in the sense of Lamport's famous essay [Lam12]. In such a proof, levels can be folded and unfolded at will.

This notion of detail is the most important one for very elementary mathematics such as the mean value example in Lamport's essay. But in advanced mathematics the issues start long before checking side conditions. The first issue is to understand precisely what is claimed. Here the first obstacle come from standing assumptions that are explained very far away or left completely implicit. Then come the notations that are needed to tame complexity, often with some ambiguity trade-off. There can be also variations on terminology even for rather fundamental definitions (for instance a compact topological space may be assumed to be Hausdorff or not depending on your country or field of expertise). Then the most important source of confusion is the use of implicit isomorphisms or inclusions (ranging from the inclusion of natural numbers into rational numbers to the inclusion of topological spaces into condensed sets).

Proof assistants try very hard to let users use generic notations and leave out inclusion maps. But they do not accept a statement unless they manage to reconstruct all this information. Once it is reconstructed it is kept at hand for inspection. And of course all definitions and lemma statements are very easy to access from their use site.

Once a statement is understood, one can turn to its proof. Here the most crucial computer capability is to patiently keep ready for display the so-called "tactic state" which is a list of all objects and assumptions that are currently relevant and the current goal of the proof. This information changes at every single step of the proof, and it is simply impossible to repeat it on paper at every step. As an example, assume we have a function $f : \mathbb{R} \to \mathbb{R}$ which is continuous at a point x_0 and a sequence u converging to x_0. The goal is to prove that the sequence $f \circ u$ converges to $f(x_0)$. At the beginning of the proof, the tactic state (in the proof assistant Lean [Mou+15]) will look like this:

```
1 goal
f : ℝ → ℝ
u : ℕ → ℝ
x₀ : ℝ
hu : sequence_tendsto u x₀
hf : continuous_function_at f x₀
⊢ sequence_tendsto (f ∘ u) (f x₀)
```

The first line claims we currently have only one thing to prove (this number can increase while splitting an equivalence proof into two implications proof for instance). Then the next three lines lists the objects that are currently fixed. The next two lines list our current assumptions, named `hu` and `hf`. Those assumptions are written in a slightly weird way but still quite readable. Then the line starting with the $\vdash$ symbol is the current goal of the proof. The only unusual feature of that goal is fx_0 to denote $f(x_0)$ (computer scientists noticed a long time ago that the number of parentheses can very quickly get out of control when writing detailed mathematics so they use them only when absolutely needed). After telling the computer to fix a positive ε and name the positivity assumption ε_pos, the tactic state becomes:

```
1 goal
f : ℝ → ℝ
u : ℕ → ℝ
x₀ : ℝ
hu : sequence_tendsto u x₀
hf : continuous_function_at f x₀
ε : ℝ
ε_pos : ε > 0
⊢ ∃ (N : ℕ), ∀ (n : ℕ), n ≥ N → |f (u n) - f x₀| ≤ ε
```

Apart from tiny notational suprises (the simple implication arrow instead of $\Rightarrow$ and colons instead of $\in$), the new goal is the expected one. The next piece of information that is often hidden on paper is instantiation of variable objects when applying lemmas. It often happens when reading new mathematics that a sentence starting with: "Lemma X ensures it suffices to prove…" leaves the reader with little information about the mathematical objects to which Lemma X is applied. This already exists in elementary mathematics. For instance, a couple of steps later in the about proof, we can have the assumption $H : \forall x, |x - x_0| < \delta \Rightarrow |f(x) - f(x_0)| < \varepsilon$ and want to prove $|f(u_n) - f(x_0)| < \varepsilon$. A computer will happily take the instruction to apply H and answer with the new goal of proving $|u_n - x_0| < \delta$. The difference with a paper proof is that one can ask the computer how the bound variable x from H was instantiated, without deciding in advance whether this is obvious or not.

One shouldn't forget that all this information available once mathematics is formalized is not meant to be thrown to the reader by default. That would be overwhelming and not useful at all. Again, the idea is to have a progressive document where readers can dynamically choose where to ask for details when needed. Currently we don't have a very nice way to present this information, especially without installing the software. But this is changing and we hope to have a convincing proof of concept within a couple of years at most.

3 Teaching

Mathematicians also spend a lot time teaching mathematics. There is a clear overlap with the previous concern of explaining mathematical definitions, statements and proofs. But there is something more, living at a meta-mathematical level: we need to teach how to conceive and write proofs. Quite often, this is taught only indirectly: students are meant to learn by imitation. Worse, we sometimes inflict some kind of logic courses which are completely disconnected from mathematical practice. I'm not talking about courses in logic for its own sake which are perfectly fine. I'm rather thinking about discussing truth tables for logical connectives and pretending this has anything to do with our normal activity.

Teaching using a proof assistant has the obvious cost of putting a technological barrier to entry (learning syntax and navigating the software). But there are also great advantages. A huge advantage is the tactic state that was already described above. This is displayed interactively during proof writing and is invaluable for students to keep track of what is fixed and what is moving. This tool is especially relevant in elementary analysis with proofs manipulating ε's and δ's.

Even before seeing the tactic state evolve, the computer forces a very clear separation between forming a statement using logical connectives and quantifiers, using such a statement that is assumed to be true, and proving such a statement. This separation is often extremely fuzzy in students' writings but also sometimes on the teacher side, especially on blackboard instead of in print. The main temptation is to use symbols such as $\forall$, $\exists$, and $\Rightarrow$ as abbreviations for words that are related to those symbols but not exactly. For instance the sequence of symbols $P \Rightarrow Q$ means "If P is true then Q is true" but is often used as the abbreviation of "I know P is true hence I deduce Q is true". In that case $\Rightarrow$ is incorrectly used as an abbreviation of "hence" or "therefore". In the worst cases, it is used as a general punctuation sign in a proof. The sequence of symbols $\exists x, Px$ means the "there exists x such that Px" but is often used to mean "I claim there exists x such that Px and I fix one such x". The sequence of symbols $\forall x \in A$ is often used incorrectly as an abbreviation of "Consider any x in A" which would be the beginning of a *proof* of $\forall x \in A, Px$. All those incorrect forms have in common a fuzzy distinction between forming a statement, using a statement and proving it. It is often combined with a confusion about what is fixed and what is "moving", also known as the confusion between free and bound variables. Of course one can abuse the meaning of symbols like this and still understand what's going on, but this is not the usual case with students.

The computer won't let students provide wrong proofs or misuse logical symbols. It will also instantly tell them when a proof is done. Such instant feed back advantage is common to many uses of computers for teaching. But this does not guarantee that students will then write correct proofs on paper. Even translating a correct proof from computer to paper is difficult. One way to reduce the gap is to make sure the computer syntax is as close as possible to normal sentences. This makes it harder to talk to the computer because the syntax becomes more complicated and there is a cognitive dissonance when writing something that looks free form but actually follows a rigid syntax. But then transferring to paper is much easier. The following is a full proof for the sequential continuity example from above, written using a custom teaching syntax for Lean. It is not 100% readable without seeing the evolving tactic state or at least knowing that inequality assumptions about ε and n are automatically named ε_pos and n_ge, but this is still a lot easier to translate to a paper proof.

```
example (f : ℝ → ℝ) (u : ℕ → ℝ) (x₀ : ℝ)
  (hu : sequence_tendsto u x₀) (hf : continuous_function_at f x₀) :
sequence_tendsto (f ∘ u) (f x₀) :=
begin
 Let's prove that ∀ ε > 0, ∃ N, ∀ n ≥ N, |f (u n) - f x₀| ≤ ε,
 Fix ε > 0,
 By hf applied to ε using ε_pos we obtain δ such that
   (δ_pos : δ > 0) (Hf : ∀ (x : ℝ), |x - x₀| ≤ δ → |f x - f x₀| ≤ ε),
 By hu applied to δ using δ_pos we obtain N such that
   Hu : ∀ n ≥ N, |u n - x₀| ≤ δ,
 Let's prove that N works : ∀ n ≥ N, |f (u n) - f x₀| ≤ ε,
 Fix n ≥ N,
 By Hf applied to u n it suffices to prove |u n - x₀| ≤ δ,
 This is Hu applied to n using n_ge
end
```

Another advantage when using proof assistants is that students can get a much better feel for the alternance between phases where the structure of the goal dictates the next move and phases where some initiative is required. This comes from two sides: the tactic state display which constantly shows the current goal, and the clear separation between asserting a statement, using it and proving it which associates a clear next action for each logical symbol. It is also very easy to list instructions that involve taking some risk, either by instantiating an existential quantifier, specializing a universal quantifier or applying an implication. This makes it very easy to backtrack when students are stuck. Of course the fact that it is easy to write comments, delete dead ends or reorder steps is also useful.

4 Creating

Checking and explaining mathematics and teaching how to prove and write proofs is great, but many mathematicians feel like their most important activity is to create new mathematics. So we need to explore how formalized mathematics can enter the creation workflow.

The first way this can happen is related to checking. Having the ability to check partial progress with absolute certainty can be extremely useful to increase confidence and determination. But research is not a linear path advancing one lemma after the other until a theorem is proved. So a much more important capability is to be able to tweak definitions and statements without restarting from the beginning or progressively loosing track of what is proven under which assumptions. A computer can continuously recheck everything at each change of definition or lemma statement, (almost) instantly flagging what needs to be modified.

Then the computer can help cleaning up, for instance by flagging unused assumptions. It can also clarify which lemma depend on which other lemma and definitions, in a much more reliable way than dependency graphs extracted from LaTeX document that rely on the author explicitly using `\ref`.

We can also hope for the computer to automatically provide proofs of routine steps, or at least suggest a next step, or perform tedious verifications. The most primitive, but very important, way to do that is simply to search for already known results. Here I mean searching for very specific results, I will come back to more difficult searching below.

Getting such help to create new theorems is mostly science fiction with current day

technology. Everything described above already exists, but the total impact of formalizing while creating is currently a huge slow down, even for experienced users (except maybe for some parts of mathematics very directly related to logic). The hope is that technology will improve to make formalization much less time consuming. Here some form of artificial intelligence may help.

An interesting special case arises when formalizing someone else's research. Then it seems that even current technology is already a big help. For instance Johan Commelin claims the computer has been a great help to understand the proofs while formalizing during the liquid tensor experiment. Understanding such an advanced proof without a very detailed write up certainly involves some creativity.

While routine proof steps may or may not benefit from artificial intelligence, there is another dream that seems to *require* it: a good search engine for mathematical statements. The issue is that any given mathematical statement can be phrased in many different ways, independently of whether it is formalized or not. So we really need some kind of fuzzy matching of statements. Of course crude methods already exist, starting with the crudest one which simply look for a given set of words. But it seems that things like neural networks should perform much better. One can also hope that such systems could search for analogous statements in a given theory, or even suggest analogies between different theories (that last hope being pure science-fiction up to now).

All this technological help, already existing or fantasized, is very nice. But the main creation help may be more indirect. Formalized mathematics requires *clear thinking*. Long before checking anything, being forced to clearly state things can already be extremely useful to refuel the creation process. Again, this does not require getting rid of fuzzy thinking. It complements it. One can alternate between fuzzy phases and clear thinking, each one helping the other.

This clear thinking is part of the creation process but it can also become part of the end result. Indeed formalized mathematics encourage, or even sometimes require, *powerful abstractions*. For instance, we are all familiar with the zoo of limits in elementary analysis, including limits of sequences, limits of functions at a point, one-sided limits, limits of function at $(\pm\infty)$ etc. On paper there is not much cost to simply assume the reader could prove any required variation of all elementary lemmas about limits, such as the squeeze theorem or composition lemmas. But a quick count reveals there are (literally) thousands of such variations. So a good abstraction is required in a formalized context. Fortunately such a thing has already been invented by Bourbaki (in the somewhat semi-formalized context of the *Éléments de mathématiques*). The key here is the notion of filter. This predates the birth of proof assistant by about 30 years. But Bourbaki makes very little use of filters compared to contemporary libraries of formalized mathematics. There filters are used in a much more systematic and versatile way, see [HIH13] for the beginning of this story. For instance, the following computation is the TEXversion of the proof of Heine-Cantor (asserting that a continuous function f from a compact space X to a space Y is always uniformly continuous) in mathlib [com20],

Lean's mathematical library:

$$\begin{aligned}
(f \times f)_* \mathcal{U}_X &= (f \times f)_* \inf_x \mathcal{N}_{(x,x)} && \text{since } X \text{ is compact separated} \\
&= \inf_x \left[(f \times f)_* \mathcal{N}_{(x,x)} \right] \\
&\leq \inf_x \mathcal{N}_{(f(x),f(x))} && \text{since } f \text{ is continuous} \\
&\leq \inf_y \mathcal{N}_{(y,y)} \\
&\leq \mathcal{U}_Y
\end{aligned}$$

As in Bourbaki, the context of this theorem is the theory of uniform spaces (a common abstraction of metric spaces and topological groups), $\mathcal{U}_X$ is the uniform structure on X, seen as a filter on $X \times X$, and for every point z, $\mathcal{N}_z$ is its neighborhood filter. The above is essentially Bourbaki's proof, but turned into a computation using a lot of algebraic structure instead of lots of sentences.

This difference is partly explained by the global development of mathematics and the modern emphasis on functoriality. But I claim that the formalization process also played a very important role in this story of filters. An interesting point here is that filters were not widely adopted by mathematics after their introduction by Bourbaki. It would be very interesting to see if their use in formalized mathematics could have a bigger impact.

In more extreme cases, the computer rigidity can also completely switch from being seen as an obstacle to being a creation opportunity. For instance the computer makes it harder to ignore the existence of various "inclusion" maps (for instance the "inclusion" of natural numbers into real numbers or the "inclusion" of a metric space into its completion). It tries very hard to automatically insert them and keep them very discrete (for instance Lean denotes automatically inserted inclusion maps by tiny up pointing arrows). But it really keeps track of them and sometimes this can be inconvenient. However this also encourages carefully thinking about the important properties of those maps as well as their functorial properties. For instance while working on topological foundations during the formalization of the definition of perfectoid spaces [BCM20], this led us to a more general version of Bourbaki's extension by continuity theorem which significantly simplifies the construction of completions of general topological rings. In that case it was also a joy to see how little broke when we switched to the more general statement: the proof required almost no modification and the computer told us so immediately.

The requirements of formalized mathematics can also lead to significant simplifications in the opposite way, using less powerful results rather than more general results. On paper it is very tempting to simply quote very sophisticated results without thinking too much about whether a simpler proof exist. A spectacular example arose in the liquid tensor experiment when Johan Commelin realized that seemingly crucial uses of the so-called Breen-Deligne resolutions in condensed mathematics can be replaced with a much cheaper technical tool [Sch21]. This happened after having carefully isolated the crucial properties of Breen-Deligne resolutions that were needed for a proof, an activity which is naturally encouraged by formalized mathematics.

5 Collaborating and having fun

Hopefully, all the stories above explain why some people think formalized mathematics and proof assistants could become very useful in a not too distant future. But they do not really explain why more and more people already use this with current day technology.

I think a key point is that formalized mathematics brings a lot of fun. Part of this fun comes from the "video game" aspect of proof assistants. But, from my perspective, the real fun comes from collaboration. It is very easy to collaborate on a formalized mathematics project. The first reason is of course that perfectly precise statements are much better for coordination. Again there is no reason to permanently display this perfect precision, but it is always available. Delegating subtasks is very easy when those are perfectly specified. We've seen impressive examples of that during the liquid tensor experiment, allowing contributions from people who had no clue about the global goal of the project.

On top of that, we have continuous checking which means that anybody can contribute without fearing to subtly break something or misunderstand some standing assumption or earlier statement. The ability to reorganize a theory without fearing to introduce uncaught inconsistencies is something very new. Such a thing would have allowed Bourbaki to incorporate newer developments and we would have category theory in the *Éléments de mathématiques* for instance.

Of course fun is not the only reason why people collaborate on building libraries of formalized mathematics. This is also an exhilarating experience where contributors feel they could have a real impact on the mathematical community, without removing anything we love, only adding new possibilities.

References

[BCM20] Kevin Buzzard, Johan Commelin, and Patrick Massot. "Formalising perfectoid spaces". In: *Proceedings of the 9th ACM SIGPLAN International Conference on Certified Programs and Proofs, CPP 2020, New Orleans, LA, USA, January 20-21, 2020*. Ed. by Jasmin Blanchette and Catalin Hritcu. ACM, 2020, pp. 299–312. URL: https://doi.org/10.1145/3372885.3373830 (cit. on p. 8).

[com20] The mathlib community. "The lean mathematical library". In: *Proceedings of the 9th ACM SIGPLAN International Conference on Certified Programs and Proofs, CPP 2020, New Orleans, LA, USA, January 20-21, 2020*. Ed. by Jasmin Blanchette and Catalin Hritcu. ACM, 2020, pp. 367–381. URL: https://doi.org/10.1145/3372885.3373824 (cit. on p. 7).

[Gon07] Georges Gonthier. "The Four Colour Theorem: Engineering of a Formal Proof". In: *Computer Mathematics, 8th Asian Symposium, ASCM 2007, Singapore, December 15-17, 2007. Revised and Invited Papers*. 2007, p. 333 (cit. on p. 2).

[Hal+17] Thomas Hales et al. "A formal proof of the Kepler conjecture". In: *Forum of Mathematics, Pi* 5 (2017), e2, 29 (cit. on p. 2).

[Har21] Kevin Hartnett. *New math book rescues landmark topology proof*. Online magazine. 2021. URL: https://www.quantamagazine.org/new-math-book-rescues-landmark-topology-proof-20210909/ (cit. on p. 3).

[HIH13] Johannes Hölzl, Fabian Immler, and Brian Huffman. "Type Classes and Filters for Mathematical Analysis in Isabelle/HOL". In: *Interactive Theorem Proving - 4th International Conference, ITP 2013, Rennes, France, July 22-26, 2013. Proceedings*. 2013, pp. 279–294 (cit. on p. 7).

[Lam12] Leslie Lamport. “How to write a 21st century proof”. In: *J. Fixed Point Theory Appl.* 11.1 (2012), pp. 43–63. URL: https://lamport.azurewebsites.net/pubs/proof.pdf (cit. on p. 3).

[Mou+15] Leonardo Mendonça de Moura et al. “The Lean Theorem Prover (System Description)”. In: *Automated Deduction - CADE-25 - 25th International Conference on Automated Deduction, Berlin, Germany, August 1-7, 2015, Proceedings*. 2015, pp. 378–388 (cit. on p. 3).

[Sch20] Peter Scholze. *Liquid tensor experiment*. Blog post. 2020. URL: https://xenaproject.wordpress.com/2020/12/05/liquid-tensor-experiment/ (cit. on p. 2).

[Sch21] Peter Scholze. *Half a year of the Liquid Tensor Experiment: Amazing developments*. Blog post. 2021. URL: https://xenaproject.wordpress.com/2021/06/05/half-a-year-of-the-liquid-tensor-experiment-amazing-developments/ (cit. on pp. 2, 8).