Machine-checked mathematics

Assia Mahboubi

Assia Mahboubi
Inria
Palaiseau, France
assia.mahboubi@inria.fr

Machine-checked mathematics

Assia Mahboubi is a permanent researcher at the French research institute Inria. She also works at the Microsoft-Inria Joint Centre. She has contributed to the formalization of the Feit–Thompson theorem in the Coq proof assistant and is currently working on verified computer algebra. In this article she gives an overview about machine-checked mathematics.

Proof assistants are software tools dedicated to the conception of digital libraries of formalized mathematics, that can be *machine-checked* automatically. So far mostly used by researchers in computer science, they have recently started to attract the attention of researchers in pure mathematics, thanks to the successful formalization of recent and major mathematical results. This article discusses the issues and the benefits of this alternate way of doing mathematics with the help of a computer.

Black boards and computers

The popular representation of a researcher is usually a person wearing a white lab coat and playing with an emblematic tool of her discipline: a microscope for a biologist, glasses and test tubes for a chemist, a telescope for the astrophysicist, ... But what does a mathematics laboratory look like? Researchers in mathematics do not need lab coats nor protection glasses. But if you ask them about it, they will probably make a strong requirement about at least their office and seminar rooms being equipped with a black board, and even may be other places like the cafeteria too. There is certainly a black board – white board preference split, just like there is a coffee – tea one, but in any case these vertical surfaces play a fundamental role in the most creative and fun part of the activity of research in mathematics: understanding a proof, a definition, shaking a wannabe theorem by trying to find counter-examples, explain and explore new ideas ...

Since the last half of the twentieth century however, computers have deeply changed the face of research in mathematics. Mathematicians nowadays seldom exchange snail mails, as J.-P. Serre and A. Grothendieck [6, 19] did not so long ago, but emails — see for instance the correspondence of the two French mathematicians C. Villani and C. Mouhot, reprinted in part in C. Villani's novel *Birth of a Theorem* [22]. They write research blog posts, they typeset themselves their articles using scientific document preparation systems, and they access them via the Internet — and less and less often at the library. Computers have also relieved mathematicians from pedestrian calculation work, now handed down to scientific computing software, able to compute more than what would be possible in a life time. This easy access to heavy computations has even motivated new forms of mathematics, like computer algebra or numerical analysis. In various areas of mathematics, there are now theorems whose sole known proofs rely as of today on the execution of a computer program, and on superhuman computational resources. Famous examples include the Four Colour Theorem [1], the existence of the Lorentz attractor [21], the Kepler conjecture [10] or the weak Goldbach conjecture [12], ...

What is a theorem?

The most visible outcome of research in mathematics is the production of new results, called theorems, together with their proofs. What makes a candidate proof become a theorem is a blend of a social process and of a scrutinization: people give talks explaining their new ideas to their peers, they eventually submit papers to journals, and these submissions are carefully reviewed by anonymous colleagues. In the vast majority of cases, everything goes fine and only minor errors subsist in published texts, that can easily be fixed by the target audience of specialists. But sometimes proofs are published that are truly incomplete or essentially incorrect, and nonetheless believed to be true for a while. There are notorious long sagas of trials and errors like the Four Colour Theorem [8] or Hilbert's 16th problem [13] — although the latter example is an emblematic example of how fertile false proofs can turn out to be. It is indeed in some cases very difficult to find reviewers who are at the same time: competent enough in all the areas of expertise required, focused enough to detect all the potential flaws, and insensitive to subjective information like their personal relation to the author or his/her reputation in the com-

munity. On this topic, the interested reader may watch V. Voevodsky explaining his misfortunes in a popular science talk at the Institute of Advanced Study, Princeton (http://video.ias.edu/node/6395). We also recommend the reading of W. Thurston's reflexions [20] on the process of mathematics.

In principle, the whole mathematical literature could be expressed in a non-ambiguous formal language, like set-theory and first-order logic, and proofs could be detailed in an exhaustive manner, making verification (boring to death but) trivial and absolutely objective. But this is not how one communicates mathematics in practice. The language of mathematics is made of a complex apparatus of abstractions, ellipsis and notations which requires some culture, years of training, but which also just makes the communication of ideas possible. Providing all the seemingly missing details to an audience of specialists would not only be utterly pedantic, but in fact it would soon obfuscate the discourse completely, sterilize creativity and miss the point of what makes mathematics beautiful. The famous group of French mathematicians N. Bourbaki make it explicit in *The Architecture of Mathematics* [4]: "What the axiomatic method sets as its essential aim is exactly that which logical formalism by itself cannot provide, namely the profound intelligibility of mathematics." Clarifying the abstract structures, which capture the deep similarities shared by seemingly extremely different objects is what it is about. "To lay down the rules of this language, to set up its vocabulary and to clarify its syntax, all that is indeed extremely useful; indeed this constitutes one aspect of the axiomatic method, the one that can properly be called logical formalism (or 'logistics' as it is sometimes called). But we emphasize that it is but one aspect of this method, indeed the least interesting one." [4]. To summarize, a verbose expansion of all the definitions and arguments of a given proof, down to the initial actions of a logical foundation, would both be too boring a task and fail to help understanding its content and checking its correctness.

Photo: flickr.com, Nelson Minar

W.S. Jevons' Logic Piano

Machines can help

Except may be if one could take benefit from the super computing powers of machines... The idea of building machines in order to mechanize the process of verifying deductions is certainly an old one (see for instance G. Leibniz' Calculus Raciocinator in his 1966 doctoral thesis, or W.S. Jevons' Logic Piano built in 1869), but the advent of computers, modern logic and computer science have turned it into concrete tools that can help for real. Indeed *Proof Assistants* are pieces of software that provide an environment for defining mathematical objects, their properties and the associated candidate proofs in a formal language well suited to this purpose. There are many such proof assistants, just like there are many programming languages, computer algebra systems or typesetting softwares. But all of

them share the same general anatomy. The first ingredient of a proof assistant is the logical foundation used to fix the formal language of mathematics: although most mathematicians are used to set theory and first-order logic, there are many more possible options, some being better-suited to the formalization of mathematical concepts *in practice*. The majority of available proof assistants are actually based on an instance of *type theory*, instead of set theory, and allow quantification on arbitrary objects and functions, instead of being limited to first-order logic. The grammatical constructions and rules of this language are rigid enough so that a program called the *kernel* of the proof assistant can be used to check the correctness of proofs written by the user, by means of a purely mechanical automated process. The kernel is really the cornerstone of a proof assistant: if one trusts this single piece of code, one trusts all the proofs it validates. Formalizing mathematics with a proof assistant means forging *by hand* a formal description of the mathematical definitions, theorem statements and candidate proofs: only the verification of the latter is automated, and not its discovery. Therefore the main difficulty is to bridge the gap between the very low level language that allows for this routine verification and the much higher-level language that make mathematics intelligible to humans. This gap is akin to the distance between the language in which human beings write programs and the patterns of bits corresponding to the physical commands executed by the processor. The bulk of proof assistants is thus to provide tools that help with this matter. The code implementing these tools is carefully separated from the one of the kernel, so that the trusted base of code remains clearly identified.

Formalizing mathematics

Proof assistants have been around for about half a century now, since the pioneering work of N. G. de Bruijn [16] on his Automath system (a project he started in 1967). These systems have been mostly used by computer science inclined users, concerned with obtaining the highest possible confidence in the behavior of *programs*. Describing the properties of the output of an algorithm and the behavior of the program implementing this algorithm in a given language is indeed a notoriously difficult task. Impressive successes have been obtained in this area in the last decade like the verification of the core of an operating system [14] and the implementation of a certified compiler for (a subset of) the C programming language [15]. Another flagship application of formal proofs is the verification of mathematical results that depend on heavy calculations. Some major mathematical results of this kind have also been machine-checked using proof assistants, like the Four Colour Theorem [8] or Hales' proof of the Kepler conjecture [11], the latter completed after over a decade of collaborative work under the direction of Th. Hales. Proof assistants and formal proofs have nonetheless failed so far to catch the interest of the majority of researchers in mathematics. One of the reasons that may explain this situation is a general feeling that the technology is not yet mature enough. In particular, a blatant lack of significant libraries of digitized mathematics has long prevented interested external users to consider starting a formalization project of their own research results, to the notable exception of Th. Hales work on his formal proof of the Kepler conjecture. Formalizing the proof of a theorem with a proof assistant remains immensely more time-consuming than writing it down with a scientific word processing software. And to the best of our knowledge, no existing proof assistant can pretend to offer a digital formal library covering a comprehensive graduate curriculum in pure mathematics.

A formal proof of the Odd Order Theorem

Indeed, finding the most appropriate methodology and techniques to craft formal definitions of mathematical objects in type theory and to construct reusable formal libraries remains to a large extent a research topic. These issues motivated the work of a team of researchers led by G. Gonthier on the formal verification of a landmark result in finite group theory due to W. Feit and J. G. Thompson, called the Odd Order Theorem [7]. This project uses a proof assistant called the Coq system and took six years to be completed [9]. The algebraic structure of group is an abstract object which is at the same time very simple, and very deep: F. Klein, author of the seminal Erlangen research program, defined geometry as the study of properties that remain invariant under the action of a given group of transformations. *Simple groups* are the most elementary, atomic instances of groups and any other finite group can be built from the finite simple ones — like molecules from these atoms. The classification of finite simple groups describes precisely the shape that a finite simple group can take, for each possible cardinal, and the Odd Order Theorem is a corner stone of this edifice:

Theorem [Odd Order Theorem]. *Every finite group of odd order is solvable*.

As a direct consequence, the Odd Order Theorem actually provides a complete description of the structure of simple groups with an odd cardinal: they are necessarily cyclic. The simplicity of the statement of this result strikingly contrasts with the sophistication of its proof, which calls for a combination of arguments of local analysis and of character theory of finite groups. The original proof [7] was published as a 250 page paper, a record at the time, and was later extensively polished and revised [3, 17]. R. Solomon describes its impact by saying that: "This short sentence and its long proof were a moment in the evolution of finite group theory analogous to the emergence of fish onto dry land. Nothing like it had happened before; nothing quite like it has happened since." [18] The validity of the Odd Order Theorem itself has never been really called into question, but the story of the classification of finite simple groups has been

The author's Odd Order Theorem bookshelf

```
Section JacobsonDensity.

Variables (gT : finGroupType) (G : {group gT}) (n : nat).
Variable rG : mx_representation F G n.
Hypothesis irrG : mx_irreducible rG.

Local Notation E_G := (enveloping_algebra_mx rG).
Local Notation Hom_G := 'C(E_G)%MS.

Lemma mx_Jacobson_density : ('C(Hom_G) <= E_G)%MS.
Proof.
apply/row_subP=> iB; rewrite -[row iB _]vec_mxK; move defB: (vec_mx _) => B.
have{defB} cBcE: (B \in 'C(Hom_G))%MS by rewrite -defB vec_mxK row_sub.
have rGnP: mx_repr G (fun x => lin_mx (mulmxr (rG x)) : 'A_n).
  split=> [|x y Gx Gy]; apply/row_matrixP=> i.
    by rewrite !rowE mul_rV_lin repr_mx1 /= !mulmx1 vec_mxK.
  by rewrite !rowE mulmxA !mul_rV_lin repr_mxM //= mxvecK mulmxA.
move def_rGn: (MxRepresentation rGnP) => rGn.
pose E_Gn := enveloping_algebra_mx rGn.
pose e1 : 'rV[F]_(n ^ 2) := mxvec 1%:M; pose U := cyclic_mx rGn e1.
have U_e1: (e1 <= U)%MS by rewrite cyclic_mx_id.
have modU: mxmodule rGn U by rewrite cyclic_mx_module.
pose Bn : 'M_(n ^ 2) := lin_mx (mulmxr B).
suffices U_e1Bn: (e1 *m Bn <= U)%MS.
  rewrite mul_vec_lin /= mul1mx in U_e1Bn; apply: submx_trans U_e1Bn _.
  rewrite genmxE; apply/row_subP=> i; rewrite row_mul rowK mul_vec_lin_row.
  by rewrite -def_rGn mul_vec_lin /= mul1mx (eq_row_sub i) ?rowK.
have{cBcE} cBncEn A: centgmx rGn A -> A *m Bn = Bn *m A.
  rewrite -def_rGn => cAG; apply/row_matrixP; case/mxvec_indexP=> j k /=.
  rewrite !rowE !mulmxA -mxvec_delta -(mul_delta_mx (0 : 'I_1)).
  rewrite mul_rV_lin mul_vec_lin /= -mulmxA; apply: (canLR vec_mxK).
  apply/row_matrixP=> i; set dj0 := delta_mx j 0.
  pose Aij := row i \o vec_mx \o mulmxr A \o mxvec \o mulmx dj0.
  have defAij := mul_rV_lin1 [linear of Aij]; rewrite /= {2}/Aij /= in defAij.
  rewrite -defAij row_mul -defAij -!mulmxA (cent_mxP cBcE) {k}//.
  rewrite memmx_cent_envelop; apply/centgmxP=> x Gx; apply/row_matrixP=> k.
  rewrite !row_mul !rowE !{}defAij /= -row_mul mulmxA mul_delta_mx.
  congr (row i _); rewrite -(mul_vec_lin (mulmxr_linear _ _)) -mulmxA.
  by rewrite -(centgmxP cAG) // mulmxA mx_rV_lin.
suffices redGn: mx_completely_reducible rGn 1%:M.
  have [V modV defUV] := redGn _ modU (submx1 _); move/mxdirect_addsP=> dxUV.
```

A piece of Coq code

much more uneven and controversial, with famous skeptics like J.P. Serre [5]. As of today, the confidence of mathematicians in this complex edifice still rests more upon the reputation of the rare experts who are able to understand this composite proof in extenso than on a genuine assimilation by the community.

Teaching proof assistants

The first and foremost motivation of the formalization was not to track petty errors and holes in this proof. The interest of this proof as a case study for the formalization of mathematics, beside the significance of the result, is the broad spectrum of algebraic theories in its prerequisite, like graduate-level linear and multilinear algebra, Galois theory, representation theory and character theory of finite groups, constructions of algebraic closures,... As of today, this realization constitutes the largest corpus of algebraic theories ever formalized. The challenge was hence to represent formally all the mathematical objects that play a role in this proof: starting from the definition of natural numbers, and covering the 250 pages of the proof, plus all the pre-requisite in-between.

One of the main issues in formalization is to model the aforementioned training the mathematician went through to be able to make sense of a mathematical text. Consider for instance the sentence:

> The determinant of a square matrix $A \in M_n(R)$ is $\mathrm{Det}(\mathrm{A}) = \sum_{\sigma \in S_n} \epsilon_\sigma \prod_i a_{\sigma(i),i}$.

The average reader is able to infer without thinking about it that the Greek letters Σ and Π, notations for iterated binary operations on the domain described in subscript, here apply respectively to the addition and to the multiplication of the ring structure which equips R, that the iteration domain of the product is necessarily $1 \leq i \leq n$ and that the signature ϵ_σ should be understood as its embedding in R. Although none of these details is explicitly denoted in the formula, they can all be deduced from the context of the formula. The information seemingly missing in such a string of symbols is however necessary for the computer to make sense of the statement. Yet it would not be reasonable to expect the user of the proof assistant to provide it by hand by decorating of the words constituting the formal sentence. (S)he would soon no more see the forest for the trees. Fortunately the training of the reader can be modeled by implementing and running appropriate algorithms which are able to infer the canonical properties hidden behind standard notations, once these rules and habits have been explicited completely. For this purpose, it is very helpful to work with a formal language based on a flavor of *type* theory. As in usual programming languages, types are labels carrying information like the domain and codomain loci of functions. Well-formed sentences satisfy constraints on their types, which are prescribed by the rules of the formal language. This helps structuring the sentences and ruling out nonsensical ones, hence facilitating the verification. But not all type annotations need to be provided by the user, as some of them can be retrieved by just enforcing the constraints on the types. This mechanism, called *type inference*, also comes from the theory of programming languages and can be used to reconstruct what the trained reader can read behind the notations, by encoding enough information in the types. This has been extensively used in the libraries of the Odd Order Theorem formal proof to preserve the readability of statements, the modularity of the theories and the ability to superimpose several views on a same object. In the end, the code producing the formula

$$\sum_{\sigma \in S_n} \epsilon_\sigma \prod_i a_{\sigma(i),i}$$

in the LaTeX typesetting language is:

```
\sum_{\sigma \in S_n}\epsilon_
{\sigma}\prod_i a_{\sigma (i),i}
```

when an analogue in Coq would be:

```
Definition det (R : ringType) n
(A : 'M[R]_n) : R := \sum_(s : 'S_n)
(-1) ^+ s * \prod_i A i (s i).
```

For a more detailed insight into the use of type inference in the formalization of mathematics, the interested reader can refer to J. Avigad's introduction [2].

Computers and black boards

Writing machine-checked digital libraries of formalized mathematics is not like using a super spell-checker for mathematics, nor does it intend to superseed the work of human reviewers — although it will hopefully assist them in this task. Formalizing mathematics is a creative research activity, not because it produces new, previously unknown theorems, but because it sparks off a reflection on the most appropriate formal representation of mathematical objects, which allows for the coherence of the involved theories and provides an appropriate socle for the upcoming layers of formalization. Revisiting mathematical theories with the help of a proof assistant is not only about chasing errors in proofs: it helps exploring and improving the architecture of a proof and the layout of the theories at stake, with the precious help of a computer. Proof assistants can help teaching mathematics, they could foster collaborative work and may be one day allow the discovery of new objects, of useful abstractions. Although the technology of proof assistants is actually already mature enough to serve the purpose of the verification of any theorem, all the components of their skeleton are still active topics of research: the logical foundations that should provide the most convenient framework for the mechanization of proofs, the automation tools helping best the users, the best user interface,... Hopefully proof assistants will rapidly evolve towards as widely used tools as computer algebra systems and eventually change the reviewing standards in mathematical journals. But there is little doubt that black (or white) board will remain the most vital piece in a mathematician's office.

References

1 Kenneth Appel and Wolfgang Haken, *Every Planar Map is Four Colorable*, Contemporary Mathematics, Vol. 98, American Mathematical Society, 1989. With the collaboration of J. Koch.

2 Jeremy Avigad, Type inference in mathematics, *Bull. Eur. Assoc. Theor. Comput. Sci. EATCS* 106 (2012), 78–98.

3 Helmut Bender and Georges Glauberman, *Local Analysis for the Odd Order Theorem*, London Mathematical Society, LNS, Vol. 188, Cambridge University Press, 1994.

4 Nicolas Bourbaki, The architecture of mathematics, *The American Mathematical Monthly* 57(4), 1950.

5 C.T. Chong and Y.K. Leong, An interview with Jean-Pierre Serre, *Math. Intelligencer* 8(4) (1986), 8–13.

6 Pierre Colmez and Jean-Pierre Serre, eds., *Grothendieck–Serre correspondence*, Bilingual Edition, American Mathematical Society – Societe Mathematique de France, 2004.

7 Walter Feit and John G. Thompson, Solvability of groups of odd order, *Pacic Journal of Mathematics* 13(3) (1963), 775–1029.

8 Georges Gonthier, Formal proofIthe four-color theorem, *Notices Amer. Math. Soc.* 55(11) (2008), 1382–1393.

9 Georges Gonthier et al., A machine-checked proof of the odd order theorem, in *Interactive Theorem Proving*, Lecture Notes in Comput. Sci., Vol. 7998, Springer, 2013, pp. 163–179.

10 Thomas C. Hales, A proof of the Kepler conjecture, *Ann. of Math. (2)* 162(3) (2005), 1065–1185.

11 Thomas C. Hales et al., A formal proof of the Kepler conjecture, arxiv.org/abs/1501.02155, January 2015.

12 Harald A. Helfgott, The ternary Goldbach conjecture is true, arxiv.org/abs/1312.7748, 2013.

13 Yu. Ilyashenko, Centennial history of Hilbert's 16th problem, *Bull. Amer. Math. Soc. (N.S.)* 39(3) (2002), 301–354.

14 Gerwin Klein et al., sel4: formal verification of an os kernel, in *SOPS ACM SIGOPS*, 2009, pp. 207–220.

15 Xavier Leroy, Formal certification of a compiler back-end, or: programming a compiler with a proof assistant, in *POPL*, ACM Press, 2006, pp. 42–54.

16 R.P. Nederpelt, J.H. Geuvers and R.C. de Vrijer, eds., *Selected Papers on Automath*, Studies in Logic and the Foundations of Mathematics, Vol. 133, North-Holland, 1994.

17 Thomas Peterfalvi, *Character Theory for the Odd Order Theorem*, London Mathematical Society, LNS, Vol. 272, Cambridge University Press, 2000.

18 Ronald Solomon, A brief history of the classification of the finite simple groups, *Bull. Amer. Math. Soc. (N.S.)* 38(3) (2001), 315–352.

19 John Tate, Correspondance Grothendieck–Serre, *Nieuw Archief voor Wiskunde* 5/5(1) (2004), 42–44.

20 William P. Thurston, On proof and progress in mathematics, *Bull. Amer. Math. Soc. (N.S.)* 30(2) (1994), 161–177.

21 Warwick Tucker, The Lorenz attractor exists, *Comptes Rendus de l'Academie des Sciences, Series I – Mathematics* 328(12) (1999), 1197–1202.

22 Cedric Villani, *Birth of a Theorem: A Mathematical Adventure*, Farrar, Straus and Giroux, April 2015.