The Mechanization of Mathematics

Jeremy Avigad

Department of Philosophy and
Department of Mathematical Sciences
Carnegie Mellon University

May 2018

Prelude

In 1998, Thomas Hales announced a proof of the Kepler conjecture.

The result relied on extensive computation.

He found the review process at the *Annals* unsatisfying:

- It was four years before they began their work.
- They cautioned that they could not check the results.

In response, he launched the *Flyspeck* project to verify the results formally.

Prelude

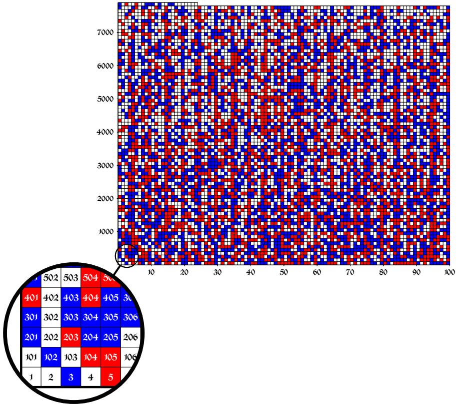
Ronald Graham posed the *Pythagorean triples problem*:

> *Is it possible to color the positive integers red and blue such that there is no monochromatic pythagorean triple* $(a^2 + b^2 = c^2)$*?*

In 2016, Marijn Heule, Oliver Kullmann, and Victor Marek showed:

- There is such a coloring of the integers from 1 to 7,824.
- There is no such coloring of the integers from 1 to 7,825.

Prelude



They used a propositional satisfiability solver for this purpose.

The proof of the negative result is 200 terabytes long.

Prelude

Computers are commonly used in mathematics:

- numerical methods in applied mathematics
- computer algebra systems

Some uses of computers in pure mathematics:

- the four color theorem (Appel and Haken, 1974)
- the existence of the Lorenz attractor (Tucker, 2002)
- the 290 theorem (Bhargava and Hanke, 2002)
- a bound on exceptional slopes in Thurston's Dehn surgery theorem (Lackenby and Meyerhoff, 2013)
- . . . and many others.

Prelude

But the results just described have a different character:

- Flyspeck was dedicated to *verification*.
- For the Pythagorean triple problem, the computer was used to carry out a heuristic search.
- In the negative case, the result was a formal proof.

In other words, they rely on the use of *formal methods*.

That is what this talk is about.

The mechanization of mathematics

This talk is based on a survey that will appear in the June/July issue of *Notices of the AMS*.

It will be published as an opinion piece.

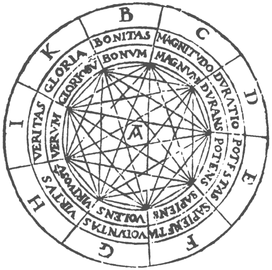
Outline of this talk:

- historical background
- formal methods in computer science
- verified proof
- verified computation
- formal search
- digital infrastructure

I'll present some opinions and conclusions at the end.

Historical origins

In his *Ars generalis ultima*, Ramon Llull, a thirteenth century Francisan monk, presented logical and visual aids to win Muslims over to the Christian faith.



Historical origins

Key insights:

- We can represent concepts, assertions, or objects of thought with symbolic tokens.
- Compound concepts (or assertions, or thoughts) can be obtained by forming combinations of more basic ones.
- Mechanical devices, even as simple as a series of concentric wheels, can be helpful in constructing and reasoning about such combinations.

Historical origins

Llull's ideas were an inspiration to Gottfried Leibniz, who dubbed the method *ars combinatoria*.

His 1666 treatise, *Dissertatio de arte combinatoria*, contained a mixture of logic and modern combinatorics.

In this treatise, Leibniz famously proposed the development of a *characteristica universalis* and a *calculus ratiocinator*.

Formal methods in computer science

Formal methods are used for

- specifying,
- developing, and
- verifying

complex hardware and software systems.

They rely on:

- *formal languages* to make assertions and express constraints,
- *formal semantics* to specify intended meaning, and
- *formal rules of inference* to verify claims and carry out search.

Formal methods in computer science

In short, they are used to

- say things,
- find things,
- and check things.

Examples:

- Model checkers search for counterexamples to specifications.
- Interactive theorem provers show that hardware and software designs meet their specifications

Formal methods in computer science

- Intel and AMD use ITP to verify processors.
- Microsoft uses formal tools to verify programs and drivers.
- CompCert has verified the correctness of a C compiler.
- The seL4 microkernel has been verified.
- Airbus uses formal methods to verify avionics software.
- Toyota uses formal methods for hybrid systems to verify control systems.
- Formal methods were used to verify Paris' driverless line 14 of the Metro.
- The NSA uses (it seems) formal methods to verify cryptographic algorithms.
- Aesthetic Integration uses formal methods to that trading software complies with regulations.

Formal methods in computer science

There is no sharp line between industrial and mathematical verification:

- Designs and specifications are expressed in mathematical terms.
- Claims rely on background mathematical knowledge.

Here I will focus on mathematics:

- Problems are conceptually deeper, less homogeneous.
- More user interaction is needed.

Formal methods in mathematics

I will discuss four domains of application:

- verified proof
- verified computation
- formal search
- digital infrastructure

Verified proof

Interactive Theorem Proving provides one method of verifying mathematical theorems.

Working with a proof assistant, users construct a formal axiomatic proof.

In most systems, this proof object can be extracted and verified independently.

Verified proof

Some systems with substantial mathematical libraries:

- Mizar (set theory)
- HOL (simple type theory)
- Isabelle (simple type theory)
- HOL light (simple type theory)
- Coq (constructive dependent type theory)
- ACL2 (primitive recursive arithmetic)
- PVS (classical dependent type theory)
- Agda (constructive dependent type theory)
- Metamath (set theory)
- Lean (dependent type theory)

Verified proof

Some theorems formalized to date:

- the prime number theorem
- the four-color theorem
- the Jordan curve theorem
- Gödel's first and second incompleteness theorems
- Dirichlet's theorem on primes in an arithmetic progression
- the central limit theorem

There are good libraries for elementary number theory, real and complex analysis, point-set topology, measure-theoretic probability, abstract algebra, Galois theory, . . .

Verified proof

Georges Gonthier and coworkers verified the Feit-Thompson Odd Order Theorem in Coq.

- The original 1963 journal publication ran 255 pages.
- The formal proof is constructive.
- The development includes libraries for finite group theory, linear algebra, and representation theory.

The project was completed on September 20, 2012, with roughly

- 150,000 lines of code,
- 4,000 definitions, and
- 13,000 lemmas and theorems.

Verified proof

Thomas Hales announced the completion of the formal verification of the Kepler conjecture (*Flyspeck*) in August 2014.

- Most of the proof was verified in HOL light.
- The classification of tame graphs was verified in Isabelle.
- Verifying several hundred nonlinear inequalities required roughly 5000 processor hours on the Microsoft Azure cloud.

Verified proof

Homotopy Type Theory provides a synthetic approach to algebraic topology.

- Constructive dependent type theory has natural homotopy-theoretic interpretations (Voevodsky, Awodey and Warren).
- Types are interpreted as spaces.
- For a, b of type A, $a = b$ says there is a path from a to b.
- Rules for equality support common patterns of reasoning about paths.
- One can consistently add an axiom to the effect that "equivalent structures are identical."

This makes it possible to reason "homotopically" in systems based on dependent type theory.

Verified proof

```
theorem PrimeNumberTheorem:
  "(%n. pi n * ln (real n) / (real n)) ----> 1"

!C. simple_closed_curve top2 C ==>
  (?A B. top2 A /\ top2 B /\
    connected top2 A /\ connected top2 B /\
  ~(A = EMPTY) /\ ~(B = EMPTY) /\
   (A INTER B = EMPTY) /\ (A INTER C = EMPTY) /\
       (B INTER C = EMPTY) /\
      (A UNION B UNION C = euclid 2)

!d k. 1 <= d /\ coprime(k,d)
  ==> INFINITE { p | prime p /\ (p == k) (mod d) }
```

Verified proof

```
Theorem Sylow's_theorem :
  [/\ forall P,
      [max P | p.-subgroup(G) P] = p.-Sylow(G) P,
      [transitive G, on 'Syl_p(G) | 'JG],
      forall P, p.-Sylow(G) P ->
        #|'Syl_p(G)| = #|G : 'N_G(P)|
   &  prime p -> #|'Syl_p(G)| %% p = 1%N].

Theorem Feit_Thompson (gT : finGroupType)
   (G : {group gT}) :
  odd #|G| → solvable G.

Theorem simple_odd_group_prime (gT : finGroupType)
    (G : {group gT}) :
  odd #|G| → simple G → prime #|G|.
```

Verified proof

```
theorem (in prob_space) central_limit_theorem:
  fixes X :: "nat ⇒ 'a ⇒ real"
    and μ :: "real measure"
    and σ c :: real
    and S :: "nat ⇒ 'a ⇒ real"
  assumes X_indep: "indep_vars (λi. borel) X UNIV"
    and X_integrable: "⋀n. integrable M (X n)"
    and X_mean: "⋀n. expectation (X n) = c"
    and σ_pos: "σ > 0"
    and X_square_integrable:
        "⋀n. integrable M (λx. (X n x)^2)"
    and X_variance: "⋀n. variance (X n) = σ^2"
    and X_distrib: "⋀n. distr M borel (X n) = μ"
  defines "S n x ≡ ∑i<n. X i x"
  shows "weak_conv_m (λn. distr M borel
       (λx. (S n x - n * c) / sqrt (n*σ^2)))
    std_normal_distribution"
```

Verified computation

Flyspeck required verifying the results of mathematical computation.

Question: what does that mean?

Some approaches:

- rewrite the computations to construct proofs as well (Flyspeck: nonlinear bounds)
- verify certificates (e.g. proof sketches, duality in linear programming)
- verify the algorithm, and then execute it with a specialized (trusted) evaluator (Four color theorem)
- verify the algorithm, extract code, and run it (trust a compiler or interpreter) (Flyspeck: enumeration of tame graphs)

Verified computation

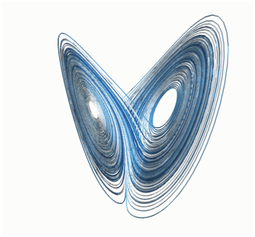
Frédéric Chyzak, Assia Mahboubi, Thomas Sibut-Pinote, and Enrico Tassi verified Apéry's 1973 proof of the irrationality of $\zeta(3)$.

- Maple worksheets by Bruno Salvy carried out symbolic computation.
- The group exported enough information to construct proofs in Coq.
- Extra work was required to handle side conditions ignored by Maple.

Verified computation

To prove the existence of the Lorenz attractor, Warwick Tucker:

- enclosed a Poincaré section with small rectangles
- used careful numerics to show that each rectangle (and associated cone) is mapped to another such.



Fabian Immler:

- formalized dynamical systems in Isabelle
- defined data structures e.g. for affine arithmetic
- verified the computation above

Verified computation

Henry Cohen and Noam Elkies in “New Upper Bounds On Sphere Packings. I”:

> *These bounds were calculated using a computer. However, the mathematics behind the calculations is rigorous. In particular, we use exact rational arithmetic, and apply Sturm’s theorem to count real roots and make sure we do not miss any sign changes.*

Marc Lackenby and Robert Meyerhoff in “The Maximal Number of Exceptional Dehn Surgeries”:

> *We now discuss computational issues and responses arising from our parameter space analysis. The computer code was written in C++.*

Verified computation

Manjul Bhargava and Jonathan Hanke In their preprint on "Universal Quadratic Forms and the 290 Theorem":

> *As with any large computation, the possibility of error is a real issue. This is especially true when using a computer, whose operation can only be viewed intermittently and whose accuracy depends on the reliability of many layers of code beneath the view of all but the most proficient computer scientist. We have taken many steps to ensure the accuracy of our computations, the most important of which are described below.*

What are referees supposed to do? We need solutions that scale.

Formal search

Automated reasoners, constraint solvers, and theorem provers implement powerful search methods.

Alas, applications to mathematics to date are few and far between.

The proof of the Pythagorean Triples Theorem by Heule, Kullman, and Marek is a striking example.

Formal search

In 1996, William McCune used an equational theorem prover to prove the Robbins conjecture, which states that a certain system of equations axiomatizes Boolean algebras.

But this was posed by a logician, Tarski.

McCune showed that $(w((x^{-1}w)^{-1}z))((yz)^{-1}y) = x$ axiomatizes groups.

Kenneth Kunen showed that this is the shortest such axiom.

Formal search

Consider a sequence $(x_i)_{i>0}$, where each x_i is ± 1.

Consider sums of this sequence along multiples of a fixed positive integer, such as

- $x_1 + x_2 + x_3 + \dots$
- $x_2 + x_4 + x_6 + \dots$
- $x_3 + x_6 + x_9 + \dots$

Erdős asked whether for some sequence these sums have a uniform bound, i.e. there is an (x_i) and C such that for every n and d,

$$|\sum_{i=1}^{n} x_{id}| \leq C.$$

Formal search

In 2014, Boris Konev and Alexei Lisitsa used a SAT solver to settle the case $C = 2$:

- there is a finite sequence $x_1, \ldots, x_{1,160}$ with discrepancy at most 2,
- but no such sequence of length 1,161.

In 2015 Terence Tao proved the full theorem with an ordinary proof (but the previous result provides a sharp bound for $C = 2$).

Formal search

Other examples:

- Heule recently showed that the Schur number $S(5)$ is equal to 160
- Heule on van der Waerden numbers
- Daniel Bundala, Michael Codish, Luís Cruz-Filipe, Peter Schneider-Kamp, Jakub Závodný on optimal sorting networks
- Felix Arends, Joël Ouaknine, and Charles Wampler on Kochen-Specker systems
- Curtis Bright, Ilias Kotsireas, Wilfrid Laurier, Vijay Ganesh on Williamson Matrices

Formal search

The results so far involve fairly simple combinatorical and algebraic structures.

We still need to learn how to use such tools for core mathematics, and make some problems amenable to search methods.

For all we know, lots of lovely theorems of mathematics can only be proved that way.

Digital infrastructure

Latex, e-mail, the web, MathOverflow, MathSciNet, and so on have had a strong influence on mathematical research.

Contemporary digital technologies for

- storage,
- search, and
- communication

of mathematical information provide another market for formal methods in mathematics.

Digital infrastructure

Thomas Hales has launched the *Formal Abstracts* project to encourage mathematicians to write formal abstracts of their papers.

A seed grant from the Sloan Foundation (University of Pittsburgh, Carnegie Mellon University, Hanoi VAST and Thang Long University) will help develop the necessary infrastructure.

To process and check the definitions, we will use the *Lean* interactive theorem prover.

Our hope is that this will provide a workable and useful first step.

Summary

We have considered applications of formal methods in mathematics:

- verified proof
- verified computation
- formal search
- digital infrastructure

They are not ready yet for prime time:

- Applications to date are isolated, extreme examples.
- Tools require lots of training and effort.

But the technology can, in the long run, transform mathematics.
We need to understand how to use it.

Summary

In the short term, we need to seek out:

- domains where verification is essential (delicate patterns of reasoning, computational results)
- domains where formal tools provide short-term gains.

Opinions

From the *Notices* article:

> *The mathematics community needs to put some skin in the game. . . . Proving theorems is not like verifying software, and computer scientists do not earn promotions or secure funding by making mathematicians happy. We need to buy into the technology if we want to reap the benefits.*

The mathematical logic community in particular needs to pay more attention.

Opinions

Formal methods rely on classical results in logic:

- formal languages and axiomatic systems
- semantics, definability, and completness proofs
- structural proof theory
- computability theory
- normalization and the lambda calculus
- Skolemization and properties
- decision procedures
- model theory of algebraic structures
- Craig's interpolation lemma

Computer scientists have added many important insights, but the theory guides the enterprise.

Opinions

In the twentieth century, mathematical logic made important contributions, clarifying

- basic concepts of mathematics
- methods of proof and rules of inference
- notions of language, meaning, expressivity, and definability
- the notion of computation

These had bearing on all aspects of mathematics, as well as computer science, linguistics, philosophy, and beyond.

But what have we done lately?

Opinions

Challenges for logicians:

- Develop formal models of everyday mathematical language.
- Develop formal models of everyday mathematical proof.
- Develop proof procedures that do well on the kinds of problems that come up "in practice."
- Develop ways of verifying mathematics at an appropriate level of abstraction.
- Understand how ordinary mathematical knowledge and expertise can guide a search.
- Understand how to represent ordinary mathematical knowledge and expertise in a robust way.
- Understand how to effectively use and combine domain-specific expertise in general settings.

Conclusions

Formal methods are valuable to mathematics:

- They provide additional precision.
- They provide strong guarantees of correctness.
- They make it possible to verify mathematical computation.
- They make it possible to use automation in a reliable way.
- They assist in the storing, finding, and communicating mathematical knowledge.

This is a promising market for mathematical logic.

Conclusions

The take-home message:

- The goal of mathematics is to extend mathematical knowledge and understanding.
- Computers change the kinds of proofs that we can discover and verify.
- In the long run, formal methods will play an important role in mathematics.
- Successes to date rely on a twentieth century logical understanding of mathematical method.
- There are still fundamental conceptual problems that need to be addressed.
- Mathematical logic still has a role to play.