Formalizing Mathematical Proofs by Computer

John Harrison

Intel Corporation

15 April 2012

Summary

- I: Formalization and Computers
  - Principia Mathematica
  - Formalization in current mathematics
  - The role of computers
- II: Theorem Proving Technology
  - Theorem provers vs. computer algebra systems
  - Early research in automated reasoning
  - Interactive proof and prover architecture
- III: Applications
  - In pure mathematics
  - In computer system verification
  - The Flyspeck project

I: Formalization and Computers

100 years since Principia Mathematica

Principia Mathematica was the first sustained and successful actual formalization of mathematics.

100 years since Principia Mathematica

Principia Mathematica was the first sustained and successful actual formalization of mathematics.

- This practical formal mathematics was to forestall objections to Russell and Whitehead's 'logicist' thesis, not a goal in itself.

100 years since Principia Mathematica

Principia Mathematica was the first sustained and successful actual formalization of mathematics.

- This practical formal mathematics was to forestall objections to Russell and Whitehead's 'logicist' thesis, not a goal in itself.
- The development was difficult and painstaking, and has probably been studied in detail by very few.

100 years since Principia Mathematica

Principia Mathematica was the first sustained and successful actual formalization of mathematics.

- This practical formal mathematics was to forestall objections to Russell and Whitehead's 'logicist' thesis, not a goal in itself.
- The development was difficult and painstaking, and has probably been studied in detail by very few.
- Subsequently, the idea of actually formalizing proofs has not been taken very seriously, and few mathematicians do it today.

100 years since Principia Mathematica

Principia Mathematica was the first sustained and successful actual formalization of mathematics.

- This practical formal mathematics was to forestall objections to Russell and Whitehead's 'logicist' thesis, not a goal in itself.
- The development was difficult and painstaking, and has probably been studied in detail by very few.
- Subsequently, the idea of actually formalizing proofs has not been taken very seriously, and few mathematicians do it today.

But thanks to the rise of the computer, the actual formalization of mathematics is attracting more interest.

The importance of computers for formal proof

Computers can both help *with* formal proof and give us new reasons to be interested in it:

The importance of computers for formal proof

Computers can both help *with* formal proof and give us new reasons to be interested in it:

- Computers are expressly designed for performing formal manipulations quickly and without error, so can be used to check and partly generate formal proofs.

The importance of computers for formal proof

Computers can both help *with* formal proof and give us new reasons to be interested in it:

- Computers are expressly designed for performing formal manipulations quickly and without error, so can be used to check and partly generate formal proofs.
- Correctness questions in computer science (hardware, programs, protocols etc.) generate a whole new array of difficult mathematical and logical problems where formal proof can help.

The importance of computers for formal proof

Computers can both help *with* formal proof and give us new reasons to be interested in it:

- Computers are expressly designed for performing formal manipulations quickly and without error, so can be used to check and partly generate formal proofs.
- Correctness questions in computer science (hardware, programs, protocols etc.) generate a whole new array of difficult mathematical and logical problems where formal proof can help.

Because of these dual connections, interest in formal proofs is strongest among computer scientists, but some 'mainstream' mathematicians are becoming interested too.

Russell was an early fan of mechanized formal proof

Newell, Shaw and Simon in the 1950s developed a 'Logic Theory Machine' program that could prove some of the theorems from *Principia Mathematica* automatically.

Russell was an early fan of mechanized formal proof

Newell, Shaw and Simon in the 1950s developed a 'Logic Theory Machine' program that could prove some of the theorems from *Principia Mathematica* automatically.

> *"I am delighted to know that Principia Mathematica can now be done by machinery [...] I am quite willing to believe that everything in deductive logic can be done by machinery. [...] I wish Whitehead and I had known of this possibility before we wasted 10 years doing it by hand." [letter from Russell to Simon]*

Russell was an early fan of mechanized formal proof

Newell, Shaw and Simon in the 1950s developed a 'Logic Theory Machine' program that could prove some of the theorems from *Principia Mathematica* automatically.

> *"I am delighted to know that Principia Mathematica can now be done by machinery [...] I am quite willing to believe that everything in deductive logic can be done by machinery. [...] I wish Whitehead and I had known of this possibility before we wasted 10 years doing it by hand." [letter from Russell to Simon]*

Newell and Simon's paper on a more elegant proof of one result in PM was rejected by JSL because it was co-authored by a machine.

Formalization in current mathematics

Traditionally, we understand *formalization* to have two components, corresponding to Leibniz's *characteristica universalis* and *calculus ratiocinator*.

Formalization in current mathematics

Traditionally, we understand *formalization* to have two components, corresponding to Leibniz's *characteristica universalis* and *calculus ratiocinator*.

- Express *statements* of theorems in a formal language, typically in terms of primitive notions such as sets.

Formalization in current mathematics

Traditionally, we understand *formalization* to have two components, corresponding to Leibniz's *characteristica universalis* and *calculus ratiocinator*.

- Express *statements* of theorems in a formal language, typically in terms of primitive notions such as sets.
- Write *proofs* using a fixed set of formal inference rules, whose correct form can be checked algorithmically.

Formalization in current mathematics

Traditionally, we understand *formalization* to have two components, corresponding to Leibniz's *characteristica universalis* and *calculus ratiocinator*.

- Express *statements* of theorems in a formal language, typically in terms of primitive notions such as sets.
- Write *proofs* using a fixed set of formal inference rules, whose correct form can be checked algorithmically.

Correctness of a formal proof is an objective question, algorithmically checkable in principle.

Mathematics is reduced to sets

The explication of mathematical concepts in terms of sets is now quite widely accepted (see *Bourbaki*).

- A real number is a set of rational numbers ...
- A Turing machine is a quintuple $(\Sigma, A, \ldots)$

Statements in such terms are generally considered clearer and more objective. (Consider pathological functions from real analysis ...)

Symbolism is important

The use of symbolism in mathematics has been steadily increasing over the centuries:

> *"[Symbols] have invariably been introduced to make things easy. [. . .] by the aid of symbolism, we can make transitions in reasoning almost mechanically by the eye, which otherwise would call into play the higher faculties of the brain. [. . .] Civilisation advances by extending the number of important operations which can be performed without thinking about them." (Whitehead,* An Introduction to Mathematics*)*

Formalization is the key to rigour

Formalization now has a important conceptual role in principle:

> *"…the correctness of a mathematical text is verified by comparing it, more or less explicitly, with the rules of a formalized language." (Bourbaki,* Theory of Sets*)*
>
> *"A Mathematical proof is rigorous when it is (or could be) written out in the first-order predicate language* $L(\in)$ *as a sequence of inferences from the axioms ZFC, each inference made according to one of the stated rules." (Mac Lane,* Mathematics: Form and Function*)*

What about in practice?

Mathematicians don't use logical symbols

Variables were used in logic long before they appeared in mathematics, but logical symbolism is rare in current mathematics. Logical relationships are usually expressed in natural language, with all its subtlety and ambiguity.

Logical symbols like '$\Rightarrow$' and '$\forall$' are used *ad hoc*, mainly for their abbreviatory effect.

> *"as far as the mathematical community is concerned George Boole has lived in vain" (Dijkstra)*

Mathematicians don't do formal proofs ...

The idea of actual formalization of mathematical proofs has not been taken very seriously:

> *"this mechanical method of deducing some mathematical theorems has no practical value because it is too complicated in practice." (Rasiowa and Sikorski,* The Mathematics of Metamathematics*)*
>
> *"[...] the tiniest proof at the beginning of the Theory of Sets would already require several hundreds of signs for its complete formalization. [...] formalized mathematics cannot in practice be written down in full [...] We shall therefore very quickly abandon formalized mathematics" (Bourbaki,* Theory of Sets*)*

...and the few people that do end up regretting it

> *"my intellect never quite recovered from the strain of writing [*Principia Mathematica*]. I have been ever since definitely less capable of dealing with difficult abstractions than I was before." (Russell,* Autobiography*)*

However, now we have computers to check and even automatically generate formal proofs.

Our goal is now not so much philosphical, but to achieve a real, practical, useful increase in the precision and accuracy of mathematical proofs.

Are proofs in doubt?

Mathematical proofs are subjected to peer review, but errors often escape unnoticed.

> *"Professor Offord and I recently committed ourselves to an odd mistake (Annals of Mathematics (2) 49, 923, 1.5). In formulating a proof a plus sign got omitted, becoming in effect a multiplication sign. The resulting false formula got accepted as a basis for the ensuing fallacious argument. (In defence, the final result was known to be true.)" (Littlewood,* Miscellany*)*

A book by Lecat gave 130 pages of errors made by major mathematicians up to 1900.
A similar book today would no doubt fill many volumes.

Even elegant textbook proofs can be wrong

"The second edition gives us the opportunity to present this new version of our book: It contains three additional chapters, substantial revisions and new proofs in several others, as well as minor amendments and improvements, many of them based on the suggestions we received. It also misses one of the old chapters, about the "problem of the thirteen spheres," whose proof turned out to need details that we couldn't complete in a way that would make it brief and elegant." (Aigner and Ziegler, Proofs from the Book*)*

Most doubtful informal proofs

What are the proofs where we do in practice worry about correctness?

- Those that are just very long and involved. Classification of finite simple groups, Seymour-Robertson graph minor theorem
- Those that involve extensive computer checking that cannot in practice be verified by hand. Four-colour theorem, Hales's proof of the Kepler conjecture
- Those that are about very technical areas where complete rigour is painful. Some branches of proof theory, formal verification of hardware or software

4-colour Theorem

Early history indicates fallibility of the traditional social process:

- Proof claimed by Kempe in 1879
- Flaw only point out in print by Heaywood in 1890

Later proof by Appel and Haken was apparently correct, but gave rise to a new worry:

- How to assess the correctness of a proof where many explicit configurations are checked by a computer program?

Most worries finally dispelled by Gonthier's formal proof in Coq.

Formal verification

In most software and hardware development, we lack even *informal* proofs of correctness.

Correctness of hardware, software, protocols etc. is routinely "established" by testing.

However, exhaustive testing is impossible and subtle bugs often escape detection until it's too late.

The consequences of bugs in the wild can be serious, even deadly.

Formal verification (*proving* correctness) seems the most satisfactory solution, but gives rise to large, ugly proofs.

The FDIV bug

A great stimulus to formal verification at Intel:

- Error in the floating-point division (FDIV) instruction on some early Intel®Pentium® processors in 1994
- Very rarely encountered, but was hit by a mathematician doing research in number theory.
- Intel eventually set aside US $475 million to cover the costs of replacements.

We don't want something like that to happen again!

II: Theorem Proving Techology

Theorem provers vs. computer algebra systems

Both systems for symbolic computation, but rather different:

- Theorem provers are more logically flexible and rigorous
- CASs are generally easier to use and more efficient/powerful

Some systems like MathXpert, Theorema blur the distinction somewhat ...

Limited expressivity in CASs

Often limited to conditional equations like

$$\sqrt{x^2} = \begin{cases} x & \text{if } x \geq 0 \\ -x & \text{if } x \leq 0 \end{cases}$$

whereas using logic we can say many interesting (and highly undecidable) things

$$\forall x \in \mathbb{R}.\ \forall \epsilon > 0.\ \exists \delta > 0.\ \forall x'.\ |x - x'| < \delta \Rightarrow |f(x) - f(x')| < \epsilon$$

Unclear expressions in CASs

Consider an equation $(x^2 - 1)/(x - 1) = x + 1$ from a CAS. What does it mean?

- Universally valid identity (albeit not quite valid)?
- Identity true when both sides are defined
- Identity over the field of rational functions
- ...

Lack of rigour in many CASs

CASs often apply simplifications even when they are not strictly valid.

Hence they can return wrong results.

Consider the evaluation of this integral in Maple:

$$\int_0^\infty \frac{e^{-(x-1)^2}}{\sqrt{x}}dx$$

We try it two different ways:

An integral in Maple

```
> int(exp(-(x-t)^2)/sqrt(x), x=0..infinity);
```

$$\frac{1}{2}\,\frac{e^{-t^2}\left(-\frac{3(t^2)^{\frac{1}{4}}\pi^{\frac{1}{2}}2^{\frac{1}{2}}e^{\frac{t^2}{2}}K_{\frac{3}{4}}(\frac{t^2}{2})}{t^2}+(t^2)^{\frac{1}{4}}\pi^{\frac{1}{2}}2^{\frac{1}{2}}e^{\frac{t^2}{2}}K_{\frac{7}{4}}(\frac{t^2}{2})\right)}{\pi^{\frac{1}{2}}}$$

```
> subs(t=1,%);
```

$$\frac{1}{2}\,\frac{e^{-1}\left(-3\pi^{\frac{1}{2}}2^{\frac{1}{2}}e^{\frac{1}{2}}K_{\frac{3}{4}}(\frac{1}{2})+\pi^{\frac{1}{2}}2^{\frac{1}{2}}e^{\frac{1}{2}}K_{\frac{7}{4}}(\frac{1}{2})\right)}{\pi^{\frac{1}{2}}}$$

```
> evalf(%);
```

0.4118623312

```
> evalf(int(exp(-(x-1)^2)/sqrt(x), x=0..infinity));
```

1.973732150

Early research in automated reasoning

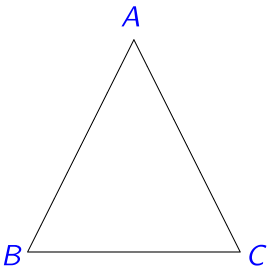
Most early theorem provers were fully automatic, even though there were several different approaches:

- Human-oriented AI style approaches (Newell-Simon, Gelerntner)
- Machine-oriented algorithmic approaches (Davis, Gilmore, Wang, Prawitz)

Modern work dominated by machine-oriented approach but some successes for AI approach.

A theorem in geometry (1)

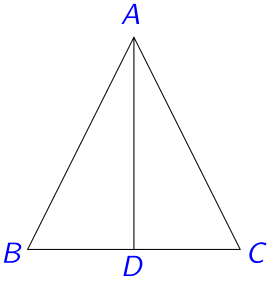
Example of AI approach in action:



If the sides AB and AC are equal (i.e. the triangle is isoseles), then the angles ABC and ACB are equal.

A theorem in geometry (2)

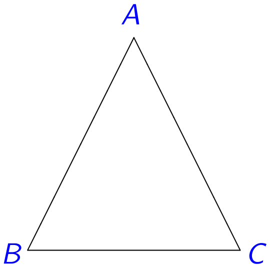
Drop perpendicular meeting BC at a point D:



and then use the fact that the triangles ABD and ACD are congruent.

A theorem in geometry (3)

Originally found by Pappus but not in many books:



Simply, the triangles ABC and ACB are congruent.

The Robbins Conjecture (1)

Huntington (1933) presented the following axioms for a Boolean algebra:

$$\begin{aligned} x + y &= y + x \\ (x + y) + z &= x + (y + z) \\ n(n(x) + y) + n(n(x) + n(y)) &= x \end{aligned}$$

Herbert Robbins conjectured that the Huntington equation can be replaced by a simpler one:

$$n(n(x + y) + n(x + n(y))) = x$$

The Robbins Conjecture (2)

This conjecture went unproved for more than 50 years, despite being studied by many mathematicians, even including Tarski.

It because a popular target for researchers in automated reasoning.

In October 1996, a (key lemma leading to) a proof was found by McCune's program EQP.

The successful search took about 8 days on an RS/6000 processor and used about 30 megabytes of memory.

What can be automated?

- Validity/satisfiability in propositional logic is decidable (SAT).
- Validity/satisfiability in many temporal logics is decidable.
- Validity in first-order logic is *semidecidable*, i.e. there are complete proof procedures that may run forever on invalid formulas
- Validity in higher-order logic is not even *semidecidable* (or anywhere in the arithmetical hierarchy).

Some specific theories

People usually use extensive background in set theory, arithmetic, algebra or geometry when they deem something 'obvious'.

- Linear theory of $\mathbb{N}$ or $\mathbb{Z}$ is decidable. Nonlinear theory not even semidecidable.
- Linear and nonlinear theory of $\mathbb{R}$ is decidable, though complexity is very bad in the nonlinear case.
- Linear and nonlinear theory of $\mathbb{C}$ is decidable. Commonly used in geometry.

Many of these naturally generalize known algorithms like linear/integer programming and Sturm's theorem.

Quantifier elimination

Many decision methods based on quantifier elimination, e.g.

- $\mathbb{C} \models (\exists x.\ x^2 + 1 = 0) \Leftrightarrow \top$
- $\mathbb{R} \models (\exists x.\ ax^2 + bx + c = 0) \Leftrightarrow a \neq 0\ \ \&\ \ b^2 \geq 4ac \vee a = 0\ \ \&\ \ (b \neq 0 \vee c = 0)$
- $\mathbb{Q} \models (\forall x.\ x < a \Rightarrow x < b) \Leftrightarrow a \leq b$
- $\mathbb{Z} \models (\exists k\ x\ y.\ ax = (5k + 2)y + 1) \Leftrightarrow \neg(a = 0)$

If we can decide variable-free formulas, quantifier elimination implies completeness.
Again generalizes known results like closure of constructible sets under projection.

Interactive theorem proving

The idea of a more 'interactive' approach was already anticipated by pioneers, e.g. Wang (1960):

> *[...] the writer believes that perhaps machines may more quickly become of practical use in mathematical research, not by proving new theorems, but by formalizing and checking outlines of proofs, say, from textbooks to detailed formalizations more rigorous that* Principia *[Mathematica], from technical papers to textbooks, or from abstracts to technical papers.*

However, constructing an effective combination is not so easy.

Who checks the checker?

Why should we believe that a formally checked proof is more reliable than a hand proof or one supported by ad-hoc programs?

- What if the underlying logic is inconsistent? Many notable logicians (Frege, Curry, Martin-Löf, ...) have proposed systems that turned out to be inconsistent.
- What if the inference rules of the logic are specified incorrectly? It's easy and common to make mistakes connected with variable capture.
- What if the proof checker has a bug? They are often large and complex pieces of software not developed to high standards of rigour

Prover architecture

The reliability of a theorem prover increases dramatically if its correctness depends only on a small amount of code.

- de Bruijn approach — generate proofs that can be certified by a simple, separate checker.
- LCF approach — reduce all rules to sequences of primitive inferences implemented by a small logical kernel.

The checker or kernel can be much simpler than the prover as a whole.
Nothing is ever certain, but we can potentially achieve very high levels of reliability in this way.

HOL Light

HOL Light is an extreme case of the LCF approach. The entire critical core is 430 lines of code:

- 10 rather simple primitive inference rules
- 2 conservative definitional extension principles
- 3 mathematical axioms (infinity, extensionality, choice)

Arguably, HOL Light is the computer-age version of *Principia*:

- The logical basis is simple type theory, which was distilled (Ramsey, Chwistek, Church) from PM's original logic.
- Everything, even arithmetic on numbers, is done from first principles by reduction to the primitive logical basis.

A simplified version of the core has itself been formally proved.

Choice of foundations

What kind of logic?

- Classical — easier and more familiar
- Constructive — natural link with computation
- Partial functions — perhaps more intuitive

What kind of mathematical framework?

- Untyped set theory
- Simple type theory
- Rich dependent type theory

Prover architecture

How to organize the construction of the prover?

- Arbitrary programming (but then how do you make it sound?)
- Based on fixed primitive inferences (the LCF approach, but you need to work hard to implement some derived rules)
- Extensible by reflection principles (prove new inference rules correct then add them to the system, which is a nice idea but very hard work)

Proof style

Directly invoking the primitive or derived rules tends to give proofs that are *procedural*. This can be quite compact and efficient.

But in some ways a *declarative* style (*what* is to be proved, not *how*) is more attractive: easier to understand independent of the prover.

Mizar pioneered the declarative style of proof, and it is now being adopted in some other systems.

There is still no consensus on what is best. Perhaps we need to be able to combine both?

A few notable general-purpose theorem provers

Different systems with various strengths and weaknesses:

- ACL2
- Coq
- HOL (HOL Light, HOL4, ProofPower, HOL Zero)
- IMPS
- Isabelle
- Mizar
- Nuprl
- PVS

See Freek Wiedijk's book *The Seventeen Provers of the World* (Springer-Verlag lecture notes in computer science volume 3600) for descriptions of many systems and a proof in each that $\sqrt{2}$ is irrational.

III: Applications

Recent formal proofs in pure mathematics

Three notable recent formal proofs in pure mathematics:

- Prime Number Theorem — Jeremy Avigad et al (Isabelle/HOL), John Harrison (HOL Light)
- Jordan Curve Theorem — Tom Hales (HOL Light), Andrzej Trybulec et al. (Mizar)
- Four-colour theorem — Georges Gonthier (Coq)

These indicate that highly non-trivial results are within reach. However these all required months/years of work.

Recent formal proofs in computer system verification

Some successes for verification using theorem proving technology:

- Microcode algorithms for floating-point division, square root and several transcendental functions on Intel® Itanium® processor family (John Harrison, HOL Light)
- CompCert verified compiler from significant subset of the C programming language into PowerPC assembler (Xavier Leroy et al., Coq)
- Designed-for-verification version of L4 operating system microkernel (Gerwin Klein et al., Isabelle/HOL).

Again, these indicate that complex and subtle computer systems can be verified, but significant manual effort was needed, perhaps tens of person-years for L4.

Some challenges and open problems

Such successes are notable, but also indicate some challenges:

- Improving level of automation so that users don't have to spend too much of their time working on essentially 'trivial' or 'obvious' lemmas.
- Incorporating results from computer calculations or symbolic computations into formal proofs in a sound but efficient way.
- Formalizing highly intuitive reasoning that is difficult to represent straightforwardly in logical deductions.

The Kepler conjecture

The *Kepler conjecture* states that no arrangement of identical balls in ordinary 3-dimensional space has a higher packing density than the obvious 'cannonball' arrangement.
Hales, working with Ferguson, arrived at a proof in 1998:

- 300 pages of mathematics: geometry, measure, graph theory and related combinatorics, ...
- 40,000 lines of supporting computer code: graph enumeration, nonlinear optimization and linear programming.

Hales submitted his proof to *Annals of Mathematics* ...

The response of the reviewers

After a full four years of deliberation, the reviewers returned:

> *"The news from the referees is bad, from my perspective. They have not been able to certify the correctness of the proof, and will not be able to certify it in the future, because they have run out of energy to devote to the problem. This is not what I had hoped for.*
> *Fejes Toth thinks that this situation will occur more and more often in mathematics. He says it is similar to the situation in experimental science — other scientists acting as referees can't certify the correctness of an experiment, they can only subject the paper to consistency checks. He thinks that the mathematical community will have to get used to this state of affairs."*

The birth of Flyspeck

Hales's proof was eventually published, and no significant error has been found in it. Nevertheless, the verdict is disappointingly lacking in clarity and finality.

As a result of this experience, the journal changed its editorial policy on computer proof so that it will no longer even try to check the correctness of computer code.

Dissatisfied with this state of affairs, Hales initiated a project called *Flyspeck* to completely formalize the proof.

Flyspeck

Flyspeck = 'Formal Proof of the Kepler Conjecture'.

> *"In truth, my motivations for the project are far more complex than a simple hope of removing residual doubt from the minds of few referees. Indeed, I see formal methods as fundamental to the long-term growth of mathematics. (Hales,* The Kepler Conjecture*)*

The formalization effort has been running for a few years now with a significant group of people involved, some doing their PhD on Flyspeck-related formalization.
In parallel, Hales has simplified the informal proof using ideas from Marchal, significantly cutting down on the formalization work.

Flyspeck: current status

- Almost all the ordinary mathematics has been formalized in HOL Light: Euclidean geometry, measure theory, *hypermaps*, *fans*, results on packings.
- Many of the linear programs have been verified in Isabelle/HOL by Steven Obua. Alexey Solovyev has recently developed a faster HOL Light formalization.
- The graph enumeration process has been verified (and improved in the process) by Tobias Nipkow in Isabelle/HOL
- Some initial work by Roland Zumkeller on nonlinear part using Bernstein polynomials. Solovyev has been working on formalizing this in HOL Light.