Computer Proofs
Where we are and where we're going

John Harrison

Intel Corporation

29th October 2013 (11:00–12:00)

Summary of talk

- Motivation for computer proof
  - The Kepler conjecture proof
  - Bugs in computer systems
- Principia Mathematica vs. current mathematics
  - Principia Mathematica
  - Formalization in current mathematics
  - The computer changes everything
- Early history and taxonomy
  - Early successes
  - Automated and interactive provers
- Current achievements
  - The world of interactive theorem provers
  - Formalized theorems and libraries of mathematics
  - The four-colour theorem
  - The odd order theorem
  - Flyspeck status
  - Univalent foundations
- The future

Motivation

Motivation from mathematics: 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.

Motivation from mathematics: 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, consisting of 300 pages of mathematics plus 40,000 lines of supporting computer code: graph enumeration, nonlinear optimization and linear programming.

Motivation from mathematics: 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, consisting of 300 pages of mathematics plus 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.

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.

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.

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" = "Formal proof of the Kepler Conjecture"

Motivation from the computer industry: the FDIV bug

One of the most serious problems that Intel has ever encountered:

- Error in the floating-point division (FDIV) instruction on some early Intel®Pentium® processors

Motivation from the computer industry: the FDIV bug

One of the most serious problems that Intel has ever encountered:

- Error in the floating-point division (FDIV) instruction on some early Intel®Pentium® processors
- Very rarely encountered, but was hit by a mathematician doing research in number theory.

Motivation from the computer industry: the FDIV bug

One of the most serious problems that Intel has ever encountered:

- Error in the floating-point division (FDIV) instruction on some early Intel®Pentium® processors
- 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.

Motivation from the computer industry: the FDIV bug

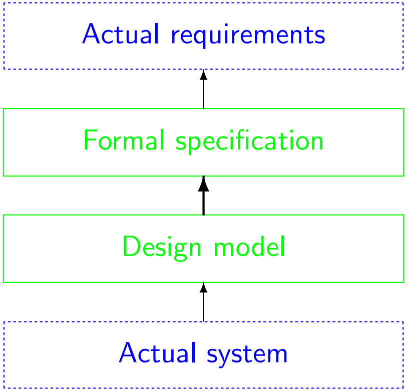
One of the most serious problems that Intel has ever encountered:

- Error in the floating-point division (FDIV) instruction on some early Intel®Pentium® processors
- 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.

A very powerful motivation for performing rigorous proofs of numerical algorithms!

Formal verification

Formal verification: mathematically prove the correctness of a *design* with respect to a mathematical *formal specification*, using machine-checked proof.



Formalization and current mathematics

Principia Mathematica

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

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.

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.

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.

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*)*

. . . Poincaré's had a particular aversion . . .

I see in logistic only shackles for the inventor. It is no aid to conciseness — far from it, and if twenty-seven equations were necessary to establish that 1 is a number, how many would be needed to prove a real theorem? If we distinguish, with Whitehead, the individual x, the class of which the only member is x and [...] the class of which the only member is the class of which the only member is x [...], do you think these distinctions, useful as they may be, go far to quicken our pace?

. . . 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*)*

. . . 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.

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.

Early history and taxonomy

Early research in automated reasoning

Most early theorem provers were fully automatic, with two main styles:

Early research in automated reasoning

Most early theorem provers were fully automatic, with two main styles:

- Human-oriented AI style approaches (Newell-Simon, Gelerntner)

Early research in automated reasoning

Most early theorem provers were fully automatic, with two main styles:

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

Early research in automated reasoning

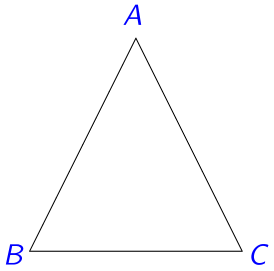
Most early theorem provers were fully automatic, with two main styles:

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

Modern work is dominated by machine-oriented approach but there have been some successes for the AI approach.

A theorem in geometry (1)

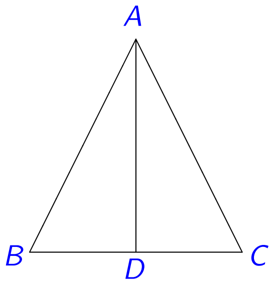
Example of AI approach in action:



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)

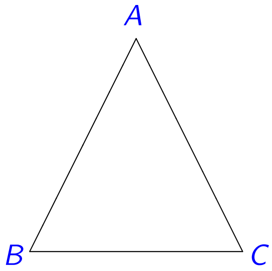
Pick bisector D of the line BC:



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.

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.

- $\mathbb{C} \models (\exists x.\ x^2 + 1 = 0) \Leftrightarrow \top$
- $\mathbb{R} \models (\exists x.\ ax^2 + bx + c = 0) \Leftrightarrow a \neq 0 \wedge b^2 \geq 4ac \vee a = 0 \wedge (b \neq 0 \vee c = 0)$

If we can decide variable-free formulas, quantifier elimination implies completeness.
Again relates to 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?

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.

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.

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.

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.

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.

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.

Current achievements

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.

Formalized theorems and libraries of mathematics

Interactive provers have been used to check quite non-trivial results, albeit not close to today's research frontiers, e.g.

- Jordan Curve Theorem — Tom Hales (HOL Light), Andrzej Trybulec et al. (Mizar)
- Prime Number Theorem — Jeremy Avigad et al (Isabelle/HOL), John Harrison (HOL Light)
- Dirichlet's Theorem — John Harrison (HOL Light)

Formalized theorems and libraries of mathematics

Interactive provers have been used to check quite non-trivial results, albeit not close to today's research frontiers, e.g.

- Jordan Curve Theorem — Tom Hales (HOL Light), Andrzej Trybulec et al. (Mizar)
- Prime Number Theorem — Jeremy Avigad et al (Isabelle/HOL), John Harrison (HOL Light)
- Dirichlet's Theorem — John Harrison (HOL Light)

According to the *Formalizing 100 theorems* page, 88% of a list of the 'top 100 mathematical theorems' have been formalized using interactive theorem provers.

In the process, provers are building up ever-larger libraries of pre-proved theorems that can be deployed in future proofs.

The four-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

The four-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?

The four-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?

In 2005, Georges Gonthier formalized the entire proof in Coq, making use of the "SSReflect" proof language and replacing ad-hoc programs by evaluation within the logical kernel.

The odd-order theorem

The odd-order theorem

- The fact that every finite group of odd order is solvable was a landmark result proved by Feit and Thompson in 1963.

The odd-order theorem

- The fact that every finite group of odd order is solvable was a landmark result proved by Feit and Thompson in 1963.
- At the time it was one of the longest mathematical proofs ever published, and it plays a major part in the full classification of simple groups.

The odd-order theorem

- The fact that every finite group of odd order is solvable was a landmark result proved by Feit and Thompson in 1963.
- At the time it was one of the longest mathematical proofs ever published, and it plays a major part in the full classification of simple groups.
- In 2012 a team led by Georges Gonthier completed a formalization in Coq, consisting of about $150,000$ lines of code.

The odd-order theorem

- The fact that every finite group of odd order is solvable was a landmark result proved by Feit and Thompson in 1963.
- At the time it was one of the longest mathematical proofs ever published, and it plays a major part in the full classification of simple groups.
- In 2012 a team led by Georges Gonthier completed a formalization in Coq, consisting of about $150,000$ lines of code.
- A fairly extensive library of results in algebra was developed in the process, including Galois theory and group characters.

The odd-order theorem

- The fact that every finite group of odd order is solvable was a landmark result proved by Feit and Thompson in 1963.
- At the time it was one of the longest mathematical proofs ever published, and it plays a major part in the full classification of simple groups.
- In 2012 a team led by Georges Gonthier completed a formalization in Coq, consisting of about $150,000$ lines of code.
- A fairly extensive library of results in algebra was developed in the process, including Galois theory and group characters.
- Uses the "SSReflect" proof language for Coq that was used in the four-colour proof.

Flyspeck: current status

A large team effort led by Hales has brought Flyspeck close to completion:

Flyspeck: current status

A large team effort led by Hales has brought Flyspeck close to completion:

- Essentially all the ordinary mathematics has been formalized in HOL Light: Euclidean geometry, measure theory, *hypermaps*, *fans*, results on packings.

Flyspeck: current status

A large team effort led by Hales has brought Flyspeck close to completion:

- Essentially all the ordinary mathematics has been formalized in HOL Light: Euclidean geometry, measure theory, *hypermaps*, *fans*, results on packings.
- The graph enumeration process has been verified (and improved in the process) by Tobias Nipkow in Isabelle/HOL.

Flyspeck: current status

A large team effort led by Hales has brought Flyspeck close to completion:

- Essentially all the ordinary mathematics has been formalized in HOL Light: Euclidean geometry, measure theory, *hypermaps*, *fans*, results on packings.
- The graph enumeration process has been verified (and improved in the process) by Tobias Nipkow in Isabelle/HOL.
- A highly optimized way of formally proving the linear programming part in HOL Light has been developed by Alexey Solovyev, following earlier work by Steven Obua.

Flyspeck: current status

A large team effort led by Hales has brought Flyspeck close to completion:

- Essentially all the ordinary mathematics has been formalized in HOL Light: Euclidean geometry, measure theory, *hypermaps*, *fans*, results on packings.
- The graph enumeration process has been verified (and improved in the process) by Tobias Nipkow in Isabelle/HOL.
- A highly optimized way of formally proving the linear programming part in HOL Light has been developed by Alexey Solovyev, following earlier work by Steven Obua.
- A method has been developed by Alexey Solovyev to prove all the nonlinear optimization results, though it still needs a lot of runtime to solve them all.

Univalent Foundations

- Provers already use quite a variety of foundations, including variants of ZFC set theory (Mizar), higher-order logic (HOL and relatives), and constructive type theory (Coq).

Univalent Foundations

- Provers already use quite a variety of foundations, including variants of ZFC set theory (Mizar), higher-order logic (HOL and relatives), and constructive type theory (Coq).
- Vladimir Voevodsky proposed a new "Homotopy Type Theory" to give 'univalent' foundations for mathematics, based on relations between homotopy and type theory.

Univalent Foundations

- Provers already use quite a variety of foundations, including variants of ZFC set theory (Mizar), higher-order logic (HOL and relatives), and constructive type theory (Coq).
- Vladimir Voevodsky proposed a new “Homotopy Type Theory” to give ‘univalent’ foundations for mathematics, based on relations between homotopy and type theory.
- In some sense it allows isomorphic objects to be identified, formalizing an intuitive principle often used by mathematicians.

Univalent Foundations

- Provers already use quite a variety of foundations, including variants of ZFC set theory (Mizar), higher-order logic (HOL and relatives), and constructive type theory (Coq).
- Vladimir Voevodsky proposed a new “Homotopy Type Theory” to give ‘univalent’ foundations for mathematics, based on relations between homotopy and type theory.
- In some sense it allows isomorphic objects to be identified, formalizing an intuitive principle often used by mathematicians.
- Voevodsky has led a major research effort resulting in new results, implementations in Coq and Agda, and a textbook.

Univalent Foundations

- Provers already use quite a variety of foundations, including variants of ZFC set theory (Mizar), higher-order logic (HOL and relatives), and constructive type theory (Coq).
- Vladimir Voevodsky proposed a new “Homotopy Type Theory” to give ‘univalent’ foundations for mathematics, based on relations between homotopy and type theory.
- In some sense it allows isomorphic objects to be identified, formalizing an intuitive principle often used by mathematicians.
- Voevodsky has led a major research effort resulting in new results, implementations in Coq and Agda, and a textbook.

An encouraging feature of both Flyspeck and Univalent Foundations is that *the driving force behind each one is a major mainstream mathematician*.

The future

What needs to change for computer proof to be taken up by the mathematical community at large, not just a few brave pioneers like Hales and Voevodsky?

The future

What needs to change for computer proof to be taken up by the mathematical community at large, not just a few brave pioneers like Hales and Voevodsky?

- Improving level of automation so that users don't have to spend too much of their time working on essentially 'trivial' or 'obvious' lemmas.

The future

What needs to change for computer proof to be taken up by the mathematical community at large, not just a few brave pioneers like Hales and Voevodsky?

- Improving level of automation so that users don't have to spend too much of their time working on essentially 'trivial' or 'obvious' lemmas.
- Developing a style of proof input that is intuitive and readable yet also concise, efficient to write and scriptable.

The future

What needs to change for computer proof to be taken up by the mathematical community at large, not just a few brave pioneers like Hales and Voevodsky?

- Improving level of automation so that users don't have to spend too much of their time working on essentially 'trivial' or 'obvious' lemmas.
- Developing a style of proof input that is intuitive and readable yet also concise, efficient to write and scriptable.
- Building up larger libraries of pre-proved mathematics so that one does not have to prove basic prerequisite apparatus from first principles.

The future

What needs to change for computer proof to be taken up by the mathematical community at large, not just a few brave pioneers like Hales and Voevodsky?

- Improving level of automation so that users don't have to spend too much of their time working on essentially 'trivial' or 'obvious' lemmas.
- Developing a style of proof input that is intuitive and readable yet also concise, efficient to write and scriptable.
- Building up larger libraries of pre-proved mathematics so that one does not have to prove basic prerequisite apparatus from first principles.
- Incorporating results from computer calculations or symbolic computations into formal proofs in a sound but efficient way.