Formalization of Mathematics for Fun and Profit

John Harrison

Intel Corporation

23rd July 2015 (14:00–15:00)

Summary of talk

- From *Principia* to the computer age
- Formalization in current mathematics
- Recent achievements in formalization
- Reliability of machine-checked proof
- Other uses of formal proofs

From *Principia* to the computer age

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.

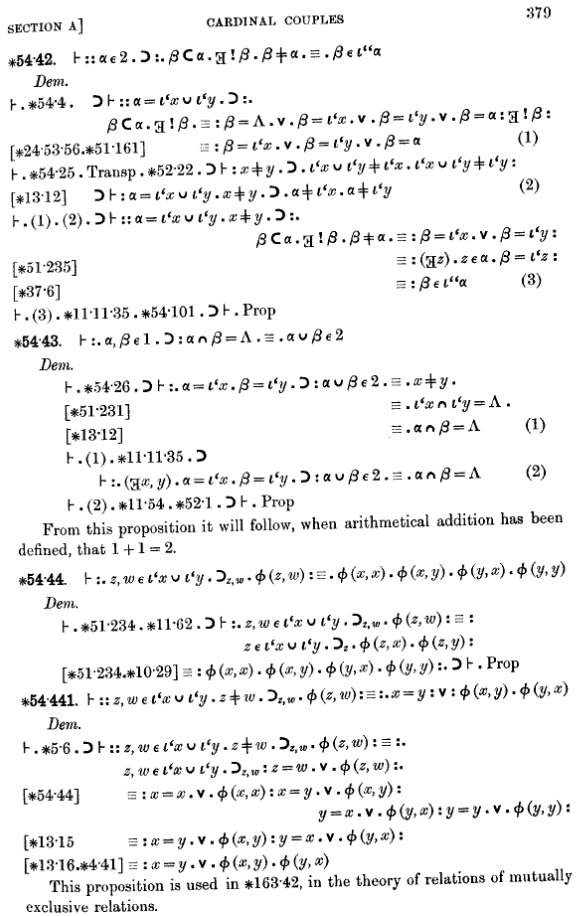
- This practical formal mathematics was to forestall objections to Russell and Whitehead's 'logicist' thesis, not a goal in itself.
- The development was difficult and painstaking, and has probably been studied in detail by very few.
- Subsequently, the idea of actually formalizing proofs has not been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

Formal proofs are difficult by hand

*"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*)*

A formal proof from 1910



SECTION A] CARDINAL COUPLES 379

$*54{\cdot}42.\quad \vdash :: \alpha \in 2 . \supset :. \beta \subset \alpha . \exists ! \beta . \beta \neq \alpha . \equiv . \beta \in \iota``\alpha$

Dem.

$\vdash . *54{\cdot}4 . \quad \supset \vdash :: \alpha = \iota`x \cup \iota`y . \supset :.$
$\beta \subset \alpha . \exists ! \beta . \equiv : \beta = \Lambda . \vee . \beta = \iota`x . \vee . \beta = \iota`y . \vee . \beta = \alpha : \exists ! \beta :$
$[*24{\cdot}53{\cdot}56.*51{\cdot}161] \quad \equiv : \beta = \iota`x . \vee . \beta = \iota`y . \vee . \beta = \alpha \qquad (1)$
$\vdash . *54{\cdot}25 . \text{Transp} . *52{\cdot}22 . \supset \vdash : x \neq y . \supset . \iota`x \cup \iota`y \neq \iota`x . \iota`x \cup \iota`y \neq \iota`y :$
$[*13{\cdot}12] \quad \supset \vdash : \alpha = \iota`x \cup \iota`y . x \neq y . \supset . \alpha \neq \iota`x . \alpha \neq \iota`y \qquad (2)$
$\vdash . (1) . (2) . \supset \vdash :: \alpha = \iota`x \cup \iota`y . x \neq y . \supset :.$
$\beta \subset \alpha . \exists ! \beta . \beta \neq \alpha . \equiv : \beta = \iota`x . \vee . \beta = \iota`y :$
$[*51{\cdot}235] \quad \equiv : (\exists z) . z \in \alpha . \beta = \iota`z :$
$[*37{\cdot}6] \quad \equiv : \beta \in \iota``\alpha \qquad (3)$
$\vdash . (3) . *11{\cdot}11{\cdot}35 . *54{\cdot}101 . \supset \vdash . \text{Prop}$

$*54{\cdot}43.\quad \vdash :. \alpha, \beta \in 1 . \supset : \alpha \cap \beta = \Lambda . \equiv . \alpha \cup \beta \in 2$

Dem.

$\vdash . *54{\cdot}26 . \supset \vdash :. \alpha = \iota`x . \beta = \iota`y . \supset : \alpha \cup \beta \in 2 . \equiv . x \neq y .$
$[*51{\cdot}231] \quad \equiv . \iota`x \cap \iota`y = \Lambda .$
$[*13{\cdot}12] \quad \equiv . \alpha \cap \beta = \Lambda \qquad (1)$
$\vdash . (1) . *11{\cdot}11{\cdot}35 . \supset$
$\vdash :. (\exists x, y) . \alpha = \iota`x . \beta = \iota`y . \supset : \alpha \cup \beta \in 2 . \equiv . \alpha \cap \beta = \Lambda \qquad (2)$
$\vdash . (2) . *11{\cdot}54 . *52{\cdot}1 . \supset \vdash . \text{Prop}$

From this proposition it will follow, when arithmetical addition has been defined, that $1 + 1 = 2$.

$*54{\cdot}44.\quad \vdash :. z, w \in \iota`x \cup \iota`y . \supset_{z,w} . \phi(z,w) : \equiv . \phi(x,x) . \phi(x,y) . \phi(y,x) . \phi(y,y)$

Dem.

$\vdash . *51{\cdot}234 . *11{\cdot}62 . \supset \vdash :. z, w \in \iota`x \cup \iota`y . \supset_{z,w} . \phi(z,w) : \equiv :$
$z \in \iota`x \cup \iota`y . \supset_z . \phi(z,x) . \phi(z,y) :$
$[*51{\cdot}234.*10{\cdot}29] \equiv : \phi(x,x) . \phi(x,y) . \phi(y,x) . \phi(y,y) :. \supset \vdash . \text{Prop}$

$*54{\cdot}441.\quad \vdash :: z, w \in \iota`x \cup \iota`y . z \neq w . \supset_{z,w} . \phi(z,w) : \equiv :. x = y : \vee : \phi(x,y) . \phi(y,x)$

Dem.

$\vdash . *5{\cdot}6 . \supset \vdash :: z, w \in \iota`x \cup \iota`y . z \neq w . \supset_{z,w} . \phi(z,w) : \equiv :.$
$z, w \in \iota`x \cup \iota`y . \supset_{z,w} : z = w . \vee . \phi(z,w) :.$
$[*54{\cdot}44] \quad \equiv : x = x . \vee . \phi(x,x) : x = y . \vee . \phi(x,y) :$
$y = x . \vee . \phi(y,x) : y = y . \vee . \phi(y,y) :$
$[*13{\cdot}15 \quad \equiv : x = y . \vee . \phi(x,y) : y = x . \vee . \phi(y,x) :$
$[*13{\cdot}16.*4{\cdot}41] \equiv : x = y . \vee . \phi(x,y) . \phi(y,x)$

This proposition is used in $*163{\cdot}42$, in the theory of relations of mutually exclusive relations.

This is p379 of Whitehead and Russell's *Principia Mathematica*.

Zooming in . . .

$*54{\cdot}43.\quad \vdash :. \alpha, \beta \in 1 . \supset : \alpha \cap \beta = \Lambda . \equiv . \alpha \cup \beta \in 2$

Dem.

$\vdash . *54{\cdot}26 . \supset \vdash :. \alpha = \iota`x . \beta = \iota`y . \supset : \alpha \cup \beta \in 2 . \equiv . x \neq y .$

[*51·231] $\equiv . \iota`x \cap \iota`y = \Lambda .$

[*13·12] $\equiv . \alpha \cap \beta = \Lambda \qquad (1)$

$\vdash . (1) . *11{\cdot}11{\cdot}35 . \supset$

$\vdash :. (\exists x, y) . \alpha = \iota`x . \beta = \iota`y . \supset : \alpha \cup \beta \in 2 . \equiv . \alpha \cap \beta = \Lambda \qquad (2)$

$\vdash . (2) . *11{\cdot}54 . *52{\cdot}1 . \supset \vdash .$ Prop

From this proposition it will follow, when arithmetical addition has been defined, that $1 + 1 = 2$.

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.

A formal proof from 2010

```
let PNT = prove
 (`(((\n. &(CARD {p | prime p /\ p <= n}) / (&n / log(&n)))
    ---> &1) sequentially`,
  REWRITE_TAC[PNT_PARTIAL_SUMMATION] THEN
  REWRITE_TAC[SUM_PARTIAL_PRE] THEN
  REWRITE_TAC[GSYM REAL_OF_NUM_ADD; SUB_REFL; CONJUNCT1 LE] THEN
  SUBGOAL_THEN `{p | prime p /\ p = 0} = {}` SUBST1_TAC THENL
   [REWRITE_TAC[EXTENSION; IN_ELIM_THM; NOT_IN_EMPTY] THEN
    MESON_TAC[PRIME_IMP_NZ];
    ALL_TAC] THEN
  REWRITE_TAC[SUM_CLAUSES; REAL_MUL_RZERO; REAL_SUB_RZERO] THEN
  MATCH_MP_TAC REALLIM_TRANSFORM_EVENTUALLY THEN
  EXISTS_TAC
   `\n. ((&n + &1) / log(&n + &1) *
         sum {p | prime p /\ p <= n} (\p. log(&p) / &p) -
         sum (1..n)
         (\k. sum {p | prime p /\ p <= k} (\p. log(&p) / &p) *
              ((&k + &1) / log(&k + &1) - &k / log(&k)))) / (&n / log(&n))` THEN
  CONJ_TAC THENL
   [REWRITE_TAC[EVENTUALLY_SEQUENTIALLY] THEN EXISTS_TAC `1` THEN SIMP_TAC[];
    ALL_TAC] THEN
  MATCH_MP_TAC REALLIM_TRANSFORM THEN
  EXISTS_TAC
   `\n. ((&n + &1) / log(&n + &1) * log(&n) -
         sum (1..n)
         (\k. log(&k) * ((&k + &1) / log(&k + &1) - &k / log(&k)))) /
        (&n / log(&n))` THEN
  REWRITE_TAC[] THEN CONJ_TAC THENL
   [REWRITE_TAC[REAL_ARITH
     `(a * x - s) / b - (a * x' - s') / b:real =
      ((s' - s) - (x' - x) * a) / b`] THEN
    REWRITE_TAC[GSYM SUM_SUB_NUMSEG; GSYM REAL_SUB_RDISTRIB] THEN
    REWRITE_TAC[REAL_OF_NUM_ADD] THEN
    MATCH_MP_TAC SUM_PARTIAL_LIMIT_ALT THEN
```

Zooming in . . .

At least the theorems are more substantial:

```
let PNT = prove
 (`(((\n. &(CARD {p | prime p /\ p <= n}) / (&n / log(&n)))
    ---> &1) sequentially`,
  REWRITE_TAC[PNT_PARTIAL_SUMMATION] THEN
  REWRITE_TAC[SUM_PARTIAL_PRE] THEN
  REWRITE_TAC[GSYM REAL_OF_NUM_ADD; SUB_REFL; CONJUNCT1 LE] THEN
```

Zooming in . . .

At least the theorems are more substantial:

```
let PNT = prove
 (`((\n. &(CARD {p | prime p /\ p <= n}) / (&n / log(&n)))
    ---> &1) sequentially`,
  REWRITE_TAC[PNT_PARTIAL_SUMMATION] THEN
  REWRITE_TAC[SUM_PARTIAL_PRE] THEN
  REWRITE_TAC[GSYM REAL_OF_NUM_ADD; SUB_REFL; CONJUNCT1 LE] THEN
```

Though whether formal proofs have become more digestible to the non-expert is perhaps questionable . . .

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

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?

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

Recent achievements in formalization

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)
- First and second Cartan Theorems — Marco Maggesi et al (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)
- First and second Cartan Theorems — Marco Maggesi et al (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.

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

A large team effort led by Hales brought Flyspeck to completion on 10th August 2014:

Flyspeck: current status

A large team effort led by Hales brought Flyspeck to completion on 10th August 2014:

- 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 brought Flyspeck to completion on 10th August 2014:

- 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 brought Flyspeck to completion on 10th August 2014:

- 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 brought Flyspeck to completion on 10th August 2014:

- 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, running in many parallel sessions of HOL Light.

Reliability of machine-checked proof

Who checks the checker?

Formalization in a proof checker is often used to ensure correctness of proofs:

- Pure mathematics — better than traditional social process
- Formal verification — often the only practical option

Who checks the checker?

Formalization in a proof checker is often used to ensure correctness of proofs:

- Pure mathematics — better than traditional social process
- Formal verification — often the only practical option

Why should we believe that these proofs are more reliable than human proofs?
What if the underlying logic is inconsistent or the proof checker is faulty?

Who cares?

The robust view:

Who cares?

The robust view:

- Bugs in theorem provers do happen, but are unlikely to produce apparent "proofs" of real results.

Who cares?

The robust view:

- Bugs in theorem provers do happen, but are unlikely to produce apparent "proofs" of real results.
- Even the flakiest theorem provers are far more reliable than most human hand proofs.

Who cares?

The robust view:

- Bugs in theorem provers do happen, but are unlikely to produce apparent "proofs" of real results.
- Even the flakiest theorem provers are far more reliable than most human hand proofs.
- Problems in specification and modelling are more likely.

Who cares?

The robust view:

- Bugs in theorem provers do happen, but are unlikely to produce apparent "proofs" of real results.
- Even the flakiest theorem provers are far more reliable than most human hand proofs.
- Problems in specification and modelling are more likely.
- Nothing is ever 100% certain, and a foundational death spiral adds little value.

We care

The hawkish view:

We care

The hawkish view:

- There has been at least one false “proof” of a real result.

We care

The hawkish view:

- There has been at least one false "proof" of a real result.
- It's unsatisfactory that we urge formality on others while developing provers so casually.

We care

The hawkish view:

- There has been at least one false "proof" of a real result.
- It's unsatisfactory that we urge formality on others while developing provers so casually.
- It should be beyond reasonable doubt that we do or don't have a formal proof.

We care

The hawkish view:

- There has been at least one false "proof" of a real result.
- It's unsatisfactory that we urge formality on others while developing provers so casually.
- It should be beyond reasonable doubt that we do or don't have a formal proof.
- A quest for perfection is worthy, even if the goal is unattainable.

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.
But it is still non-trivial . . .

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)

Everything, even arithmetic on numbers, is done by reduction to the primitive basis.

Still...

HOL Light does contain subtle code, e.g.

Still...

HOL Light does contain subtle code, e.g.

- Variable renaming in substitution and type instantiation

Still...

HOL Light does contain subtle code, e.g.

- Variable renaming in substitution and type instantiation
- Treatment of polymorphic types in definitions

It would still be nice to verify the core . . .

One fell swoop

We can imagine problems at several levels:

One fell swoop

We can imagine problems at several levels:

- The underlying logic is unsound or even inconsistent

One fell swoop

We can imagine problems at several levels:

- The underlying logic is unsound or even inconsistent
- The formal definitions of the inference rules are incorrect

One fell swoop

We can imagine problems at several levels:

- The underlying logic is unsound or even inconsistent
- The formal definitions of the inference rules are incorrect
- The implementing code contains bugs

One fell swoop

We can imagine problems at several levels:

- The underlying logic is unsound or even inconsistent
- The formal definitions of the inference rules are incorrect
- The implementing code contains bugs

To eliminate all of these:

> *Formalize the intended set-theoretic semantics of the logic and prove that the code implements inference rules that are sound w.r.t. this semantics.*

The HOL in HOL project

Project to verify an implementation of HOL Light using HOL itself (either HOL Light or HOL4) has recently been brought to completion:

The HOL in HOL project

Project to verify an implementation of HOL Light using HOL itself (either HOL Light or HOL4) has recently been brought to completion:

- Basic verification of approximation to HOL inside itself, minus definitional principles (Harrison)

The HOL in HOL project

Project to verify an implementation of HOL Light using HOL itself (either HOL Light or HOL4) has recently been brought to completion:

- Basic verification of approximation to HOL inside itself, minus definitional principles (Harrison)
- Extension of semantics to cover definitional principles and match actual code (Kumar)

The HOL in HOL project

Project to verify an implementation of HOL Light using HOL itself (either HOL Light or HOL4) has recently been brought to completion:

- Basic verification of approximation to HOL inside itself, minus definitional principles (Harrison)
- Extension of semantics to cover definitional principles and match actual code (Kumar)
- Implementation in CakeML with path to verified machine code implementation (Kumar, Myreen, Owens, . . .)

The HOL in HOL project

Project to verify an implementation of HOL Light using HOL itself (either HOL Light or HOL4) has recently been brought to completion:

- Basic verification of approximation to HOL inside itself, minus definitional principles (Harrison)
- Extension of semantics to cover definitional principles and match actual code (Kumar)
- Implementation in CakeML with path to verified machine code implementation (Kumar, Myreen, Owens, . . .)

However there are two apparent problems with 'HOL in HOL' . . .

Logical objections

Taken too literally, our goal is impossible:

Logical objections

Taken too literally, our goal is impossible:

- Tarski: you cannot formalize the semantics of HOL in itself

Logical objections

Taken too literally, our goal is impossible:

- Tarski: you cannot formalize the semantics of HOL in itself
- Gödel: you cannot prove the consistency of HOL in itself, unless it is in fact *inconsistent*

Logical objections

Taken too literally, our goal is impossible:

- Tarski: you cannot formalize the semantics of HOL in itself
- Gödel: you cannot prove the consistency of HOL in itself, unless it is in fact *inconsistent*

Actually prove two slightly different statements:

- $\mathrm{HOL} \vdash \mathit{Con}(\mathrm{HOL} - \{\infty\})$
- $\mathrm{HOL} + I \vdash \mathit{Con}(\mathrm{HOL})$.

Other uses of formal proofs

Other uses of formal proofs?

This perhaps goes back to Kreisel's question:

> *What more do we know if we have proved a theorem by restricted means than if we merely know that it is true?*

Possible applications of formal proofs?

Other uses of formal proofs?

This perhaps goes back to Kreisel's question:

> *What more do we know if we have proved a theorem by restricted means than if we merely know that it is true?*

Possible applications of formal proofs?

- Extracting constructive information or computational content (this was Kreisel's answer)

Other uses of formal proofs?

This perhaps goes back to Kreisel's question:

> *What more do we know if we have proved a theorem by restricted means than if we merely know that it is true?*

Possible applications of formal proofs?

- Extracting constructive information or computational content (this was Kreisel's answer)
- Use in education to provide precise explicit proofs with full detail or provide practice in formal reasoning.

Other uses of formal proofs?

This perhaps goes back to Kreisel's question:

> *What more do we know if we have proved a theorem by restricted means than if we merely know that it is true?*

Possible applications of formal proofs?

- Extracting constructive information or computational content (this was Kreisel's answer)
- Use in education to provide precise explicit proofs with full detail or provide practice in formal reasoning.
- Semantically well-founded corpus of mathematics for search, machine learning, sharing ...

Other uses of formal proofs?

This perhaps goes back to Kreisel's question:

> *What more do we know if we have proved a theorem by restricted means than if we merely know that it is true?*

Possible applications of formal proofs?

- Extracting constructive information or computational content (this was Kreisel's answer)
- Use in education to provide precise explicit proofs with full detail or provide practice in formal reasoning.
- Semantically well-founded corpus of mathematics for search, machine learning, sharing ...
- ...?

Sharing results between interactive provers

At the very least, we might hope to be able to share results between (similar?) interactive theorem provers:

Sharing results between interactive provers

At the very least, we might hope to be able to share results between (similar?) interactive theorem provers:

- hol90 $\rightarrow$ Nuprl: Howe and Felty 1997

Sharing results between interactive provers

At the very least, we might hope to be able to share results between (similar?) interactive theorem provers:

- hol90 $\rightarrow$ Nuprl: Howe and Felty 1997
- ACL2 $\rightarrow$ hol90: Staples 1999

Sharing results between interactive provers

At the very least, we might hope to be able to share results between (similar?) interactive theorem provers:

- hol90 $\rightarrow$ Nuprl: Howe and Felty 1997
- ACL2 $\rightarrow$ hol90: Staples 1999
- ACL2 $\rightarrow$ HOL4: Gordon, Hunt, Kaufmann & Reynolds 2006

Translating proofs between interactive provers

More interesting and foundational satisfying is to translate proofs:

Translating proofs between interactive provers

More interesting and foundational satisfying is to translate proofs:

- hol90 $\rightarrow$ Coq: Denney 2000

Translating proofs between interactive provers

More interesting and foundational satisfying is to translate proofs:

- hol90 $\rightarrow$ Coq: Denney 2000
- hol90 $\rightarrow$ NuPRL: Naumov, Stehr and Meseguer 2001

Translating proofs between interactive provers

More interesting and foundational satisfying is to translate proofs:

- hol90 $\rightarrow$ Coq: Denney 2000
- hol90 $\rightarrow$ NuPRL: Naumov, Stehr and Meseguer 2001
- HOL4 $\rightarrow$ Isabelle/HOL: Skalberg 2006

Translating proofs between interactive provers

More interesting and foundational satisfying is to translate proofs:

- hol90 $\rightarrow$ Coq: Denney 2000
- hol90 $\rightarrow$ NuPRL: Naumov, Stehr and Meseguer 2001
- HOL4 $\rightarrow$ Isabelle/HOL: Skalberg 2006
- HOL Light $\rightarrow$ Isabelle/HOL: Obua 2006

Translating proofs between interactive provers

More interesting and foundational satisfying is to translate proofs:

- hol90 $\rightarrow$ Coq: Denney 2000
- hol90 $\rightarrow$ NuPRL: Naumov, Stehr and Meseguer 2001
- HOL4 $\rightarrow$ Isabelle/HOL: Skalberg 2006
- HOL Light $\rightarrow$ Isabelle/HOL: Obua 2006
- Isabelle/HOL $\rightarrow$ HOL Light: McLaughlin 2006

Translating proofs between interactive provers

More interesting and foundational satisfying is to translate proofs:

- hol90 $\rightarrow$ Coq: Denney 2000
- hol90 $\rightarrow$ NuPRL: Naumov, Stehr and Meseguer 2001
- HOL4 $\rightarrow$ Isabelle/HOL: Skalberg 2006
- HOL Light $\rightarrow$ Isabelle/HOL: Obua 2006
- Isabelle/HOL $\rightarrow$ HOL Light: McLaughlin 2006
- HOL Light $\rightarrow$ Coq: Keller 2009

More comprehensive sharing

There are at least two major projects that allow sharing between HOL-like systems

More comprehensive sharing

There are at least two major projects that allow sharing between HOL-like systems

- OpenTheory (Hurd) — a general framework designed to support the transfer of theorems and proofs between HOL family provers

More comprehensive sharing

There are at least two major projects that allow sharing between HOL-like systems

- OpenTheory (Hurd) — a general framework designed to support the transfer of theorems and proofs between HOL family provers
- HOL Zero (Adams) — simple and transparent version of HOL designed as a vehicle for proof import and checking with importers from other HOLs.

More comprehensive sharing

There are at least two major projects that allow sharing between HOL-like systems

- OpenTheory (Hurd) — a general framework designed to support the transfer of theorems and proofs between HOL family provers
- HOL Zero (Adams) — simple and transparent version of HOL designed as a vehicle for proof import and checking with importers from other HOLs.

More in the spirit of the original QED vision is the Logosphere project, which uses the Twelf logical framework as the common 'metalogic':
`http://www.logosphere.org`

Using formal mathematics for machine learning

If we look at many AI fields we see a common recent trend:

> *Use general machine learning trained on huge datasets in preference to hand-crafted algorithms.*

Using formal mathematics for machine learning

If we look at many AI fields we see a common recent trend:

> *Use general machine learning trained on huge datasets in preference to hand-crafted algorithms.*

Even though there has been extensive work on ATP for 60 years, with strong links to AI research, there has been surprisingly little such work in using machine learning in theorem proving.

Using formal mathematics for machine learning

If we look at many AI fields we see a common recent trend:

> *Use general machine learning trained on huge datasets in preference to hand-crafted algorithms.*

Even though there has been extensive work on ATP for 60 years, with strong links to AI research, there has been surprisingly little such work in using machine learning in theorem proving.
Finally, mainly thanks to Josef Urban, there has been an explosion of interest in this area.

Using formal mathematics for machine learning

If we look at many AI fields we see a common recent trend:

> *Use general machine learning trained on huge datasets in preference to hand-crafted algorithms.*

Even though there has been extensive work on ATP for 60 years, with strong links to AI research, there has been surprisingly little such work in using machine learning in theorem proving.
Finally, mainly thanks to Josef Urban, there has been an explosion of interest in this area.
Paradoxically, learning benefits from the large formal libraries associated with *interactive* theorem provers, even though the more natural 'home' might seem to be *automated* theorem proving.

Machine learning and ATP on Flyspeck

First work reported on the Flyspeck corpus:

Cezary Kaliszyk and Josef Urban, Learning-Assisted Automated Reasoning with Flyspeck (2012).

Machine learning and ATP on Flyspeck

First work reported on the Flyspeck corpus:

> *Cezary Kaliszyk and Josef Urban, Learning-Assisted Automated Reasoning with Flyspeck (2012).*

Uses machine learning trained on the Flyspeck proofs to perform premise selection (identify lemmas likely to be useful), in conjunction with a battery of automated theorem proving tools.

Machine learning and ATP on Flyspeck

First work reported on the Flyspeck corpus:

> *Cezary Kaliszyk and Josef Urban, Learning-Assisted Automated Reasoning with Flyspeck (2012).*

Uses machine learning trained on the Flyspeck proofs to perform premise selection (identify lemmas likely to be useful), in conjunction with a battery of automated theorem proving tools.

> *It is shown that 39% of the 14185 theorems could be proved in a push-button mode (without any high-level advice and user interaction) in 30 seconds of real time on a fourteen-CPU workstation.*

HOL(y)Hammer

This 'HOL(y)Hammer' framework can be used online as a way of getting hints if you are stuck on a HOL Light proof:

`http://colo12-c703.uibk.ac.at/hh/`

HOL(y)Hammer

This 'HOL(y)Hammer' framework can be used online as a way of getting hints if you are stuck on a HOL Light proof:

`http://colo12-c703.uibk.ac.at/hh/`

It can sometimes find proofs that a human (this one, anyway) missed

> *Theorem* `FACE_OF_POLYHEDRON_POLYHEDRON` *states that a face of a polyhedron [...] is again a polyhedron. The HOL Light proof takes 23 lines [...] but a much simpler proof was found by the AI/ATP automation, based on [...] the* `FACE_OF_STILLCONVEX` *theorem: a face t of any convex set s is equal to the intersection of s with the affine hull of t. To finish the proof, one needs just three 'obvious' facts: Every polyhedron is convex (*`POLYHEDRON_IMP_CONVEX`*), the intersection of two polyhedra is again a polyhedron (*`POLYHEDRON_INTER`*), and affine hull is always a polyhedron (*`POLYHEDRON_AFFINE_HULL`*).*

Thank you!