Formal proof: current progress and outstanding challenges

John Harrison

Intel Corporation

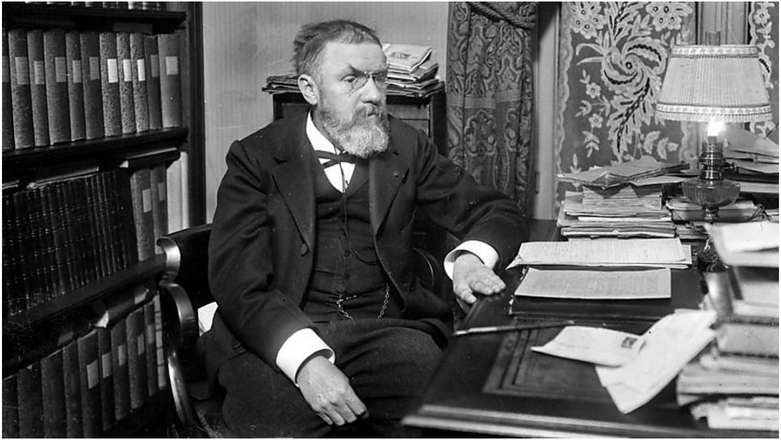
5th May 2014 (11:00–12:00)

Summary of talk

- A century of formal proof
  - Poincaré on formal proof
  - From Principia Mathematica to the computer age
  - Major milestones in formalization
  - Development of mathematical libraries
- Current perspectives
  - The provers of the world
  - Foundations
  - Software architecture
  - Proof languages
  - Automation
  - Libraries
- More about HOL Light
  - Foundations and architecture
  - Decision procedures and automation
  - A tour of the libraries
- The future

A century of formal proof

What would Poincaré have thought?



Poincaré’s had a distinct aversion to formal logic

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?

Poincaré's had a distinct aversion to formal logic

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?

However, Poincaré's was no stranger to errors

- In 1890 Poincaré's memoir on the three body problem was published in *Acta Mathematica* as the winning entry in King Oscar II's prize competition.

However, Poincaré's was no stranger to errors

- In 1890 Poincaré's memoir on the three body problem was published in *Acta Mathematica* as the winning entry in King Oscar II's prize competition.
- As a result of probing questions by Phragmén, Poincaré discovered a fundamental error *after* the prize had been awarded and the journal issue printed and even delivered to some subscribers.

However, Poincaré's was no stranger to errors

- In 1890 Poincaré's memoir on the three body problem was published in *Acta Mathematica* as the winning entry in King Oscar II's prize competition.
- As a result of probing questions by Phragmén, Poincaré discovered a fundamental error *after* the prize had been awarded and the journal issue printed and even delivered to some subscribers.
- This was a very productive mistake: the new realization led to a much deeper understanding of dynamical systems and laid the foundations of modern chaos theory.

However, Poincaré's was no stranger to errors

- In 1890 Poincaré's memoir on the three body problem was published in *Acta Mathematica* as the winning entry in King Oscar II's prize competition.
- As a result of probing questions by Phragmén, Poincaré discovered a fundamental error *after* the prize had been awarded and the journal issue printed and even delivered to some subscribers.
- This was a very productive mistake: the new realization led to a much deeper understanding of dynamical systems and laid the foundations of modern chaos theory.
- However it was embarrassing and expensive for all concerned — Poincaré spent more than the competition prize money paying for the journal issues to be recalled and reprinted.

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.

Even Russell did not enjoy doing formal proofs

*"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*)*

Even Russell did not enjoy doing formal proofs

> *"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 philosophical, 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.

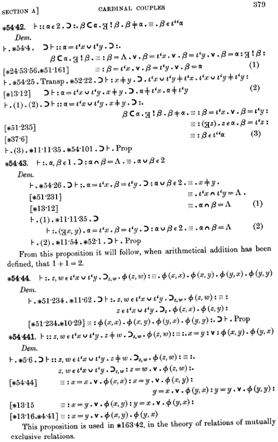
The importance of computers for formal proof

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.

A formal proof from 1910



SECTION A] CARDINAL COUPLES 379

∗54·42. $\vdash :: \alpha \in 2 \,.\supset :.\, \beta \subset \alpha \,.\, \exists ! \beta \,.\, \beta \neq \alpha \,.\equiv .\, \beta \in \iota\text{“}\alpha$

Dem.

$$\vdash .*54\cdot4 .\quad \supset \vdash :: \alpha = \iota\text{‘}x \cup \iota\text{‘}y \,.\supset :.$$
$$\beta \subset \alpha \,.\, \exists ! \beta \,.\equiv : \beta = \Lambda \,.\vee .\, \beta = \iota\text{‘}x \,.\vee .\, \beta = \iota\text{‘}y \,.\vee .\, \beta = \alpha : \exists ! \beta :$$
$$[*24\cdot53\cdot56 . *51\cdot161] \quad \equiv : \beta = \iota\text{‘}x \,.\vee .\, \beta = \iota\text{‘}y \,.\vee .\, \beta = \alpha \qquad (1)$$
$$\vdash .*54\cdot25 .\, \text{Transp} .*52\cdot22 .\supset \vdash : x \neq y \,.\supset .\, \iota\text{‘}x \cup \iota\text{‘}y \neq \iota\text{‘}x \,.\, \iota\text{‘}x \cup \iota\text{‘}y \neq \iota\text{‘}y :$$
$$[*13\cdot12] \quad \supset \vdash : \alpha = \iota\text{‘}x \cup \iota\text{‘}y \,.\, x \neq y \,.\supset .\, \alpha \neq \iota\text{‘}x \,.\, \alpha \neq \iota\text{‘}y \qquad (2)$$
$$\vdash .(1).(2).\supset \vdash :: \alpha = \iota\text{‘}x \cup \iota\text{‘}y \,.\, x \neq y \,.\supset :.$$
$$\beta \subset \alpha \,.\, \exists ! \beta \,.\, \beta \neq \alpha \,.\equiv : \beta = \iota\text{‘}x \,.\vee .\, \beta = \iota\text{‘}y :$$
$$[*51\cdot235] \quad \equiv : (\exists z) \,.\, z \in \alpha \,.\, \beta = \iota\text{‘}z :$$
$$[*37\cdot6] \quad \equiv : \beta \in \iota\text{“}\alpha \qquad (3)$$
$$\vdash .(3). *11\cdot11\cdot35 . *54\cdot101 .\supset \vdash .\, \text{Prop}$$

∗54·43. $\vdash :. \alpha, \beta \in 1 \,.\supset : \alpha \cap \beta = \Lambda \,.\equiv .\, \alpha \cup \beta \in 2$

Dem.

$$\vdash .*54\cdot26 .\supset \vdash :. \alpha = \iota\text{‘}x \,.\, \beta = \iota\text{‘}y \,.\supset : \alpha \cup \beta \in 2 \,.\equiv .\, x \neq y \,.$$
$$[*51\cdot231] \quad \equiv .\, \iota\text{‘}x \cap \iota\text{‘}y = \Lambda \,.$$
$$[*13\cdot12] \quad \equiv .\, \alpha \cap \beta = \Lambda \qquad (1)$$
$$\vdash .(1). *11\cdot11\cdot35 .\supset$$
$$\vdash :. (\exists x, y) \,.\, \alpha = \iota\text{‘}x \,.\, \beta = \iota\text{‘}y \,.\supset : \alpha \cup \beta \in 2 \,.\equiv .\, \alpha \cap \beta = \Lambda \qquad (2)$$
$$\vdash .(2). *11\cdot54 . *52\cdot1 .\supset \vdash .\, \text{Prop}$$

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

∗54·44. $\vdash :. z, w \in \iota\text{‘}x \cup \iota\text{‘}y \,.\supset_{z,w} .\, \phi(z,w) : \equiv .\, \phi(x,x) \,.\, \phi(x,y) \,.\, \phi(y,x) \,.\, \phi(y,y)$

Dem.

$$\vdash .*51\cdot234 . *11\cdot62 .\supset \vdash :. z, w \in \iota\text{‘}x \cup \iota\text{‘}y \,.\supset_{z,w} .\, \phi(z,w) : \equiv :$$
$$z \in \iota\text{‘}x \cup \iota\text{‘}y \,.\supset_z .\, \phi(z,x) \,.\, \phi(z,y) :$$
$$[*51\cdot234 . *10\cdot29] \equiv : \phi(x,x) \,.\, \phi(x,y) \,.\, \phi(y,x) \,.\, \phi(y,y) :.\supset \vdash .\, \text{Prop}$$

∗54·441. $\vdash :: z, w \in \iota\text{‘}x \cup \iota\text{‘}y \,.\, z \neq w \,.\supset_{z,w} .\, \phi(z,w) : \equiv :.\, x = y : \vee : \phi(x,y) \,.\, \phi(y,x)$

Dem.

$$\vdash .*5\cdot6 .\supset \vdash :: z, w \in \iota\text{‘}x \cup \iota\text{‘}y \,.\, z \neq w \,.\supset_{z,w} .\, \phi(z,w) : \equiv :.$$
$$z, w \in \iota\text{‘}x \cup \iota\text{‘}y \,.\supset_{z,w} : z = w \,.\vee .\, \phi(z,w) :.$$
$$[*54\cdot44] \quad \equiv : x = x \,.\vee .\, \phi(x,x) : x = y \,.\vee .\, \phi(x,y) :$$
$$y = x \,.\vee .\, \phi(y,x) : y = y \,.\vee .\, \phi(y,y) :$$
$$[*13\cdot15 \quad \equiv : x = y \,.\vee .\, \phi(x,y) : y = x \,.\vee .\, \phi(y,x) :$$
$$[*13\cdot16 . *4\cdot41] \equiv : x = y \,.\vee .\, \phi(x,y) \,.\, \phi(y,x)$$

This proposition is used in ∗163·42, in the theory of relations of mutually exclusive relations.

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

Zooming in . . .

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

Dem.

$$
\begin{aligned}
&\vdash . *54\cdot26 . \supset \vdash :. \alpha = \iota`x . \beta = \iota`y . \supset : \alpha \cup \beta \in 2 . \equiv . x \neq y . \\
&[*51\cdot231] \qquad\qquad \equiv . \iota`x \cap \iota`y = \Lambda . \\
&[*13\cdot12] \qquad\qquad \equiv . \alpha \cap \beta = \Lambda \qquad (1) \\
&\vdash . (1) . *11\cdot11\cdot35 . \supset \\
&\qquad \vdash :. (\exists x, y) . \alpha = \iota`x . \beta = \iota`y . \supset : \alpha \cup \beta \in 2 . \equiv . \alpha \cap \beta = \Lambda \qquad (2) \\
&\vdash . (2) . *11\cdot54 . *52\cdot1 . \supset \vdash . \text{Prop}
\end{aligned}
$$

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

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
  SUBGOAL_THEN `{p | prime p /\ p = 0} = {}` SUBST1_TAC THENL
```

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
  SUBGOAL_THEN `{p | prime p /\ p = 0} = {}` SUBST1_TAC THENL
```

Moreover, we can arrange to have more readable proofs — see for example Bill Richter's talk.

The major landmarks

These are arguably the three major landmarks in the formalization of mathematics

The major landmarks

These are arguably the three major landmarks in the formalization of mathematics

- The four-colour theorem (every planar map is 4-colourable) — Gonthier et al.

The major landmarks

These are arguably the three major landmarks in the formalization of mathematics

- The four-colour theorem (every planar map is 4-colourable) — Gonthier et al.
- The odd order theorem (every finite group of odd order is solvable) — Gonthier et al.

The major landmarks

These are arguably the three major landmarks in the formalization of mathematics

- The four-colour theorem (every planar map is 4-colourable) — Gonthier et al.
- The odd order theorem (every finite group of odd order is solvable) — Gonthier et al.
- The Flyspeck project (the Kepler Conjecture that no sphere packing beats face-centred cubic) — Hales et al.

The major landmarks

These are arguably the three major landmarks in the formalization of mathematics

- The four-colour theorem (every planar map is 4-colourable) — Gonthier et al.
- The odd order theorem (every finite group of odd order is solvable) — Gonthier et al.
- The Flyspeck project (the Kepler Conjecture that no sphere packing beats face-centred cubic) — Hales et al.

These are demonstrations that the technology can handle long and difficult proofs, and even that some leading mathematicians like Hales are willing to use them.

Formalized theorems and libraries of mathematics

Also important is the progress made on more modest building-blocks for mathematics, still including quite substantial results, 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)
- First and second Cartan Theorems — Marco Maggesi et al (HOL Light)

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

Current perspectives

A few notable general-purpose theorem provers

There is a diverse (perhaps too diverse?) world of proof assistants, with these being just a few:

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

A few notable general-purpose theorem provers

There is a diverse (perhaps too diverse?) world of proof assistants, with these being just a few:

- ACL2
- Agda
- Coq
- HOL (HOL Light, HOL4, ProofPower, HOL Zero)
- IMPS
- Isabelle
- Metamath
- 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 proofs that $\sqrt{2}$ is irrational.

Foundations

The choice of foundations is a difficult one, sometimes balancing simplicity against flexibility or expressiveness:

Foundations

The choice of foundations is a difficult one, sometimes balancing simplicity against flexibility or expressiveness:

- The ‘traditional’ or ‘standard’ foundation for mathematics is set theory, and some provers do use that
  - Metamath and Isabelle/ZF (standard ZF/ZFC)
  - Mizar (Tarski-Grothendieck set theory)

Foundations

The choice of foundations is a difficult one, sometimes balancing simplicity against flexibility or expressiveness:

- The 'traditional' or 'standard' foundation for mathematics is set theory, and some provers do use that
  - Metamath and Isabelle/ZF (standard ZF/ZFC)
  - Mizar (Tarski-Grothendieck set theory)
- Partly as a result of their computer science interconnections, many provers are based on type theory
  - HOL family and Isabelle/HOL (simple type theory)
  - Martin-Löf type theory (Agda, Nuprl)
  - Calculus of inductive constructions (Coq)
  - Other typed formalisms (IMPS, PVS)

Foundations

The choice of foundations is a difficult one, sometimes balancing simplicity against flexibility or expressiveness:

- The 'traditional' or 'standard' foundation for mathematics is set theory, and some provers do use that
  - Metamath and Isabelle/ZF (standard ZF/ZFC)
  - Mizar (Tarski-Grothendieck set theory)
- Partly as a result of their computer science interconnections, many provers are based on type theory
  - HOL family and Isabelle/HOL (simple type theory)
  - Martin-Löf type theory (Agda, Nuprl)
  - Calculus of inductive constructions (Coq)
  - Other typed formalisms (IMPS, PVS)
- Some are even based on very simple foundations analogous to primitive recursive arithmetic, without explicit quantifiers quantifiers (ACL2, NQTHM)

Foundations

The choice of foundations is a difficult one, sometimes balancing simplicity against flexibility or expressiveness:

- The ‘traditional’ or ‘standard’ foundation for mathematics is set theory, and some provers do use that
  - Metamath and Isabelle/ZF (standard ZF/ZFC)
  - Mizar (Tarski-Grothendieck set theory)
- Partly as a result of their computer science interconnections, many provers are based on type theory
  - HOL family and Isabelle/HOL (simple type theory)
  - Martin-Löf type theory (Agda, Nuprl)
  - Calculus of inductive constructions (Coq)
  - Other typed formalisms (IMPS, PVS)
- Some are even based on very simple foundations analogous to primitive recursive arithmetic, without explicit quantifiers quantifiers (ACL2, NQTHM)
- There is now interest in a new foundational approach, homotopy type theory, with experimental implementations.

Software architecture

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

Software 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.

Software 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.

Software 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.

Software 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.
There have even recently been papers about versions of Milawa (a simplified ACL2) and HOL Light verified right down to machine code.

Proof languages

Directly invoking the primitive or derived rules tends to give proofs that are *procedural*.

Proof languages

Directly invoking the primitive or derived rules tends to give proofs that are *procedural*.

A *declarative* style (*what* is to be proved, not *how*) can be nicer:

Proof languages

Directly invoking the primitive or derived rules tends to give proofs that are *procedural*.
A *declarative* style (*what* is to be proved, not *how*) can be nicer:

- Easier to write and understand independent of the prover

Proof languages

Directly invoking the primitive or derived rules tends to give proofs that are *procedural*.

A *declarative* style (*what* is to be proved, not *how*) can be nicer:

- Easier to write and understand independent of the prover
- Easier to modify

Proof languages

Directly invoking the primitive or derived rules tends to give proofs that are *procedural*.

A *declarative* style (*what* is to be proved, not *how*) can be nicer:

- Easier to write and understand independent of the prover
- Easier to modify
- Less tied to the details of the prover, hence more portable

Proof languages

Directly invoking the primitive or derived rules tends to give proofs that are *procedural*.

A *declarative* style (*what* is to be proved, not *how*) can be nicer:

- Easier to write and understand independent of the prover
- Easier to modify
- Less tied to the details of the prover, hence more portable
- However it can also be more verbose and less easy to script.

Proof languages

Directly invoking the primitive or derived rules tends to give proofs that are *procedural*.

A *declarative* style (*what* is to be proved, not *how*) can be nicer:

- Easier to write and understand independent of the prover
- Easier to modify
- Less tied to the details of the prover, hence more portable
- However it can also be more verbose and less easy to script.

Mizar pioneered the declarative style of proof. Recently, several other declarative proof languages have been developed, as well as declarative shells round existing systems like HOL and Isabelle.

Automation

One major obstacle to the wider use of proof assistants is the low level of automation, so it can be a struggle to prove ‘obvious’ facts. There are some quite powerful automated techniques, e.g.

Automation

One major obstacle to the wider use of proof assistants is the low level of automation, so it can be a struggle to prove 'obvious' facts. There are some quite powerful automated techniques, e.g.

- Pure logic proof search (SAT, FOL, HOL)

Automation

One major obstacle to the wider use of proof assistants is the low level of automation, so it can be a struggle to prove ‘obvious’ facts. There are some quite powerful automated techniques, e.g.

- Pure logic proof search (SAT, FOL, HOL)
- Decision procedures for numerical theories (linear arithmetic and algebra, SMT).

Automation

One major obstacle to the wider use of proof assistants is the low level of automation, so it can be a struggle to prove 'obvious' facts. There are some quite powerful automated techniques, e.g.

- Pure logic proof search (SAT, FOL, HOL)
- Decision procedures for numerical theories (linear arithmetic and algebra, SMT).
- Quantifier elimination procedures

Automation

One major obstacle to the wider use of proof assistants is the low level of automation, so it can be a struggle to prove 'obvious' facts. There are some quite powerful automated techniques, e.g.

- Pure logic proof search (SAT, FOL, HOL)
- Decision procedures for numerical theories (linear arithmetic and algebra, SMT).
- Quantifier elimination procedures

Many of these have been successfully integrated into proof assistants without compromising their soundness, e.g.

Automation

One major obstacle to the wider use of proof assistants is the low level of automation, so it can be a struggle to prove 'obvious' facts. There are some quite powerful automated techniques, e.g.

- Pure logic proof search (SAT, FOL, HOL)
- Decision procedures for numerical theories (linear arithmetic and algebra, SMT).
- Quantifier elimination procedures

Many of these have been successfully integrated into proof assistants without compromising their soundness, e.g.

- Reimplement algorithms to perform proofs as they proceed

Automation

One major obstacle to the wider use of proof assistants is the low level of automation, so it can be a struggle to prove 'obvious' facts. There are some quite powerful automated techniques, e.g.

- Pure logic proof search (SAT, FOL, HOL)
- Decision procedures for numerical theories (linear arithmetic and algebra, SMT).
- Quantifier elimination procedures

Many of these have been successfully integrated into proof assistants without compromising their soundness, e.g.

- Reimplement algorithms to perform proofs as they proceed
- Have suitable 'certificates' produced by an external tool checked in the inference kernel.

Automation

One major obstacle to the wider use of proof assistants is the low level of automation, so it can be a struggle to prove 'obvious' facts. There are some quite powerful automated techniques, e.g.

- Pure logic proof search (SAT, FOL, HOL)
- Decision procedures for numerical theories (linear arithmetic and algebra, SMT).
- Quantifier elimination procedures

Many of these have been successfully integrated into proof assistants without compromising their soundness, e.g.

- Reimplement algorithms to perform proofs as they proceed
- Have suitable 'certificates' produced by an external tool checked in the inference kernel.
- Extend kernel with verified implementation (*reflection*).

Libraries

- Another serious obstacle is the lack of libraries of ‘basic’ results, meaning that when proving a major theorem one needs constantly to be proving a stream of low-level lemmas.

Libraries

- Another serious obstacle is the lack of libraries of 'basic' results, meaning that when proving a major theorem one needs constantly to be proving a stream of low-level lemmas.
- Sometimes flashy or exciting theorems (Brouwer fixed-point theorem, the Picard theorems) aren't as useful as less showy ones (the change of variables formula for integrals etc.)

Libraries

- Another serious obstacle is the lack of libraries of ‘basic’ results, meaning that when proving a major theorem one needs constantly to be proving a stream of low-level lemmas.
- Sometimes flashy or exciting theorems (Brouwer fixed-point theorem, the Picard theorems) aren’t as useful as less showy ones (the change of variables formula for integrals etc.)
- Large formalizations (Odd Order Theorem, Flyspeck) have motivated formalization of ‘foundational’ material as a by-product, making similar efforts easier in future.

Libraries

- Another serious obstacle is the lack of libraries of ‘basic’ results, meaning that when proving a major theorem one needs constantly to be proving a stream of low-level lemmas.
- Sometimes flashy or exciting theorems (Brouwer fixed-point theorem, the Picard theorems) aren’t as useful as less showy ones (the change of variables formula for integrals etc.)
- Large formalizations (Odd Order Theorem, Flyspeck) have motivated formalization of ‘foundational’ material as a by-product, making similar efforts easier in future.
- The earliest large mathematical library, still perhaps the largest is the Mizar Mathematical Library (MML), following the style of mathematical papers with extracted text and references.

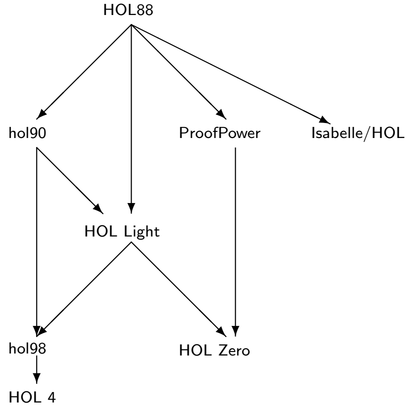
Libraries

- Another serious obstacle is the lack of libraries of ‘basic’ results, meaning that when proving a major theorem one needs constantly to be proving a stream of low-level lemmas.
- Sometimes flashy or exciting theorems (Brouwer fixed-point theorem, the Picard theorems) aren’t as useful as less showy ones (the change of variables formula for integrals etc.)
- Large formalizations (Odd Order Theorem, Flyspeck) have motivated formalization of ‘foundational’ material as a by-product, making similar efforts easier in future.
- The earliest large mathematical library, still perhaps the largest is the Mizar Mathematical Library (MML), following the style of mathematical papers with extracted text and references.
- Many theorem provers including Coq, HOL Light and Isabelle/HOL (including the ‘archive of formal proofs’) also have large and every-expanding mathematical libraries.

More about HOL Light

The HOL family DAG

There are many HOL provers, of which HOL Light is just one, all descended from Mike Gordon's original HOL system in the late 1980s.



HOL Light primitive rules (1)

$$\frac{}{\vdash t = t}\ \texttt{REFL}$$

$$\frac{\Gamma \vdash s = t \quad \Delta \vdash t = u}{\Gamma \cup \Delta \vdash s = u}\ \texttt{TRANS}$$

$$\frac{\Gamma \vdash s = t \quad \Delta \vdash u = v}{\Gamma \cup \Delta \vdash s(u) = t(v)}\ \texttt{MK\_COMB}$$

$$\frac{\Gamma \vdash s = t}{\Gamma \vdash (\lambda x.\ s) = (\lambda x.\ t)}\ \texttt{ABS}$$

$$\frac{}{\vdash (\lambda x.\ t)x = t}\ \texttt{BETA}$$

HOL Light primitive rules (2)

$$\frac{}{\{p\} \vdash p} \; \texttt{ASSUME}$$

$$\frac{\Gamma \vdash p = q \quad \Delta \vdash p}{\Gamma \cup \Delta \vdash q} \; \texttt{EQ\_MP}$$

$$\frac{\Gamma \vdash p \quad \Delta \vdash q}{(\Gamma - \{q\}) \cup (\Delta - \{p\}) \vdash p = q} \; \texttt{DEDUCT\_ANTISYM\_RULE}$$

$$\frac{\Gamma[x_1, \ldots, x_n] \vdash p[x_1, \ldots, x_n]}{\Gamma[t_1, \ldots, t_n] \vdash p[t_1, \ldots, t_n]} \; \texttt{INST}$$

$$\frac{\Gamma[\alpha_1, \ldots, \alpha_n] \vdash p[\alpha_1, \ldots, \alpha_n]}{\Gamma[\gamma_1, \ldots, \gamma_n] \vdash p[\gamma_1, \ldots, \gamma_n]} \; \texttt{INST\_TYPE}$$

Pushing the LCF approach to its limits

The main features of the LCF approach to theorem proving are:

Pushing the LCF approach to its limits

The main features of the LCF approach to theorem proving are:

- Reduce all proofs to a small number of relatively simple primitive rules

Pushing the LCF approach to its limits

The main features of the LCF approach to theorem proving are:

- Reduce all proofs to a small number of relatively simple primitive rules
- Use the programmability of the implementation/interaction language to make this practical

Pushing the LCF approach to its limits

The main features of the LCF approach to theorem proving are:

- Reduce all proofs to a small number of relatively simple primitive rules
- Use the programmability of the implementation/interaction language to make this practical

HOL Light may represent the most "extreme" application of this philosophy.

Pushing the LCF approach to its limits

The main features of the LCF approach to theorem proving are:

- Reduce all proofs to a small number of relatively simple primitive rules
- Use the programmability of the implementation/interaction language to make this practical

HOL Light may represent the most "extreme" application of this philosophy.

- HOL Light's primitive rules are very simple, and the trusted core is just a few hundred lines of code.

Pushing the LCF approach to its limits

The main features of the LCF approach to theorem proving are:

- Reduce all proofs to a small number of relatively simple primitive rules
- Use the programmability of the implementation/interaction language to make this practical

HOL Light may represent the most "extreme" application of this philosophy.

- HOL Light's primitive rules are very simple, and the trusted core is just a few hundred lines of code.
- There is an extensive suite of automated tools built on top that all reduce to this foundation.

Some of HOL Light's basic automation

- Simplifier for (conditional, contextual) rewriting.
- Tactic mechanism for mixed forward and backward proofs.
- Tautology checker.
- Automated theorem provers for pure logic, based on tableaux and model elimination.
- Linear arithmetic decision procedures over $\mathbb{R}$, $\mathbb{Z}$ and $\mathbb{N}$.
- Differentiator for real functions.
- Generic normalizers for rings and fields
- General quantifier elimination over $\mathbb{C}$
- Gröbner basis algorithm over fields

Some unusual automation

HOL Light has also introduced several novel automated proof methods, all of which were developed to answer real problems in formalization:

Some unusual automation

HOL Light has also introduced several novel automated proof methods, all of which were developed to answer real problems in formalization:

- Heuristic decision procedure for divisibility properties in number theory via a reduction to ideal membership. (For example, can prove the Chinese Remainder Theorem automatically.)

Some unusual automation

HOL Light has also introduced several novel automated proof methods, all of which were developed to answer real problems in formalization:

- Heuristic decision procedure for divisibility properties in number theory via a reduction to ideal membership. (For example, can prove the Chinese Remainder Theorem automatically.)
- Decision procedures for general 'triangle law' reasoning in normed spaces and general decision procedure for Hilbert spaces, using decidability results developed in work with Solovay and Arthan.

Some unusual automation

HOL Light has also introduced several novel automated proof methods, all of which were developed to answer real problems in formalization:

- Heuristic decision procedure for divisibility properties in number theory via a reduction to ideal membership. (For example, can prove the Chinese Remainder Theorem automatically.)
- Decision procedures for general 'triangle law' reasoning in normed spaces and general decision procedure for Hilbert spaces, using decidability results developed in work with Solovay and Arthan.
- 'Without loss of generality' tactics for simplifying goals in geometry by use of special coordinate systems, which can greatly simplify some Flyspeck goals.

A tour of the libraries (1)

Partly as a result of Flyspeck, HOL Light is particularly strong in the area of topology, analysis and geometry in Euclidean space $\mathbb{R}^n$.

File	Lines	Contents
misc.ml	562	Background stuff
vectors.ml	8627	Basic vectors, linear algebra
determinants.ml	3141	Determinant and trace
topology.ml	20235	Basic topological notions
convex.ml	11827	Convex sets and functions
paths.ml	17066	Paths, simple connectedness etc.
polytope.ml	5855	Faces, polytopes, polyhedra etc.
dimension.ml	6794	Dimensional theorems
derivatives.ml	2732	Derivatives
clifford.ml	979	Geometric (Clifford) algebra
integration.ml	17407	Integration
measure.ml	10252	Lebesgue measure

A tour of the libraries (2)

From this foundation complex analysis is developed and used to derive convenient theorems for $\mathbb{R}$ as well as more topological results.

File	Lines	Contents
complexes.ml	2036	Complex numbers
canal.ml	3760	Complex analysis
transcendentals.ml	6981	Real & complex transcendentals
realanalysis.ml	15845	Some analytical stuff on R
moretop.ml	7349	Further topological results
cauchy.ml	18231	Complex line integrals

A tour of the libraries (2)

From this foundation complex analysis is developed and used to derive convenient theorems for $\mathbb{R}$ as well as more topological results.

File	Lines	Contents
complexes.ml	2036	Complex numbers
canal.ml	3760	Complex analysis
transcendentals.ml	6981	Real & complex transcendentals
realanalysis.ml	15845	Some analytical stuff on R
moretop.ml	7349	Further topological results
cauchy.ml	18231	Complex line integrals

It would be desirable to generalize much of the material to general topological spaces, metric spaces, measure spaces etc. Some work already by Bill Richter on general topology.

Some examples from topology

The Brouwer fixed point theorem:

```
  |- !f:real^N->real^N s.
        compact s /\ convex s /\ ~(s = {}) /\
        f continuous_on s /\ IMAGE f s SUBSET s
        ==> ?x. x IN s /\ f x = x
```

The Borsuk homotopy extension theorem:

```
|- !f:real^M->real^N g s t u.
        closed_in (subtopology euclidean t) s /\
        (ANR s /\ ANR t \/ ANR u) /\
        f continuous_on t /\ IMAGE f t SUBSET u /\
        homotopic_with (\x. T) (s,u) f g
        ==> ?g'. homotopic_with (\x. T) (t,u) f g' /\
                 g' continuous_on t /\
                 IMAGE g' t SUBSET u /\
                 !x. x IN s ==> g'(x) = g(x)
```

Some examples from convexity

The Krein-Milman (Minkowski) theorem

```
|- !s:real^N->bool.
        convex s /\ compact s
        ==> s = convex hull {x | x extreme_point_of s}
```

Approximation of convex sets by polytopes w.r.t. Hausdorff distance:

```
|- !s:real^N->bool e.
        bounded s /\ convex s /\ &0 < e
        ==> ?p. polytope p /\ s SUBSET p /\
                hausdist(p,s) < e
```

Some examples from measure theory

Steinhaus's theorem:

```
|- !s:real^N->bool.
        lebesgue_measurable s /\ ~negligible s
        ==> ?d. &0 < d /\
                ball(vec 0,d) SUBSET
                {x - y | x IN s /\ y IN s}
```

Luzin's theorem:

```
|- !f:real^M->real^N s e.
        measurable s /\ f measurable_on s /\ &0 < e
        ==> ?k. compact k /\ k SUBSET s /\
                measure(s DIFF k) < e /\
                f continuous_on k
```

Some examples from complex analysis

The Little Picard theorem:

```
|- !f a b.
      f holomorphic_on (:complex) /\
      ~(a = b) /\ IMAGE f (:complex) INTER {a,b} = {}
      ==> ?c. f = \x. c
```

The Riemann mapping theorem:

```
|- !s. open s /\ simply_connected s <=>
       s = {} \/ s = (:complex) \/
       ?f g. f holomorphic_on s /\
             g holomorphic_on ball(Cx(&0),&1) /\
             (!z. z IN s
                  ==> f z IN ball(Cx(&0),&1) /\
                      g(f z) = z) /\
             (!z. z IN ball(Cx(&0),&1)
                  ==> g z IN s /\ f(g z) = z)
```

The future

Future prospects

Future prospects

- There is still lots of scope for improving automation, either with off-the-shelf methods adapted to be provably sound, or new ideas.

Future prospects

- There is still lots of scope for improving automation, either with off-the-shelf methods adapted to be provably sound, or new ideas.
- The steady increase in the stock of theorems in the prover libraries will continue and eventually make tackling a ‘typical’ mathematical problem much more tractable.

Future prospects

- There is still lots of scope for improving automation, either with off-the-shelf methods adapted to be provably sound, or new ideas.
- The steady increase in the stock of theorems in the prover libraries will continue and eventually make tackling a ‘typical’ mathematical problem much more tractable.
- New research in foundations may result in fundamentally better approaches to formalization and even have increasing influence back on mathematics itself.

Future prospects

- There is still lots of scope for improving automation, either with off-the-shelf methods adapted to be provably sound, or new ideas.
- The steady increase in the stock of theorems in the prover libraries will continue and eventually make tackling a ‘typical’ mathematical problem much more tractable.
- New research in foundations may result in fundamentally better approaches to formalization and even have increasing influence back on mathematics itself.
- Given the diversity of theorem proving systems, it seems there will be still more research into sharing and importing and exporting proofs between them.

Future prospects

- There is still lots of scope for improving automation, either with off-the-shelf methods adapted to be provably sound, or new ideas.
- The steady increase in the stock of theorems in the prover libraries will continue and eventually make tackling a 'typical' mathematical problem much more tractable.
- New research in foundations may result in fundamentally better approaches to formalization and even have increasing influence back on mathematics itself.
- Given the diversity of theorem proving systems, it seems there will be still more research into sharing and importing and exporting proofs between them.
- We can further increase the soundness guarantees by rigorous verification down to the lowest levels as well as proof checking and proof auditing.