formalization of mathematics

Freek Wiedijk
Radboud University Nijmegen

Nederlands Mathematisch Congres
Leiden University

2007 04 12, 11:45

what is formalization?

principia mathematica

- Gottlob Frege, 1879

  Begriffsschrift

  formal logic in theory

- Alfred North Whitehead & Bertrand Russell, 1910–1913

  Principia Mathematica

  formal logic in practice

  development of mathematics in a formal system

automath

- N.G. de Bruijn, 1968

  Automath

  computer makes formalization feasible

- 1971–1976

  large ZWO ($\rightsquigarrow$ NWO) project

- Bert van Benthem Jutting, 1977

  Checking Landau's 'Grundlagen' in the Automath System

  158 pages of German mathematics $\rightsquigarrow$

  491 pages of Automath source code

  checking time: couple of hours (today: under half a second)

what formalization isn't: **proofs with heavy computer support**

- Kenneth Appel & Wolfgang Haken, 1977
  four color theorem

  a good mathematical proof is like a poem –
  this is a telephone directory!

- Andrew Odlyzko & Herman te Riele, 1985
  Mertens' conjecture

  first 2000 zeroes of the Riemann zeta function to 100 decimals

- Tom Hales, 2003
  Kepler conjecture

computer only used as a calculator

what formalization isn't: **computer algebra**

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

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

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

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

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

$$0.4118623312$$

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

$$1.973732150$$

clearly no proofs are involved here

is every **Robbins algebra** a Boolean algebra?

$$a \vee b = b \vee a$$
$$a \vee (b \vee c) = (a \vee b) \vee c$$
$$\neg(\neg(a \vee b) \vee \neg(a \vee \neg b)) = a$$

EQP (by Bill McCune, Argonne National Laboratory), 1996:
'yes', with a 34 line proof

in practice automated theorem proving is almost useless

just mindless search
computers only beat humans at 'puzzles'

don't expect computers to produce interesting proofs on their own

and now, an example: **a proof by contradiction** (Mizar)

Een bolleboos riep laatst met zwier
gewapend met een vel A-vijf:
Er is geen allergrootst getal,
dat is wat ik bewijzen ga.
Stel, dat ik u nu zou bedriegen
en hier een potje stond te jokken,
dan ik zou zonder overdrijven
het grootste kunnen op gaan noemen.
Maar ben ik klaar, roept u gemeen:
‘Vermeerder dat getal met twee!’
En zien we zeker en gewis
dat dit toch niet het grootste was.
En gaan we zo nog door een poos,
dan merkt u: dit is onbegrensd.
En daarmee heb ik q.e.d.
Ik ben hier diep gelukkig door.
‘Zo gaan’, zei hij voor hij bezwijmde,
‘bewijzen uit het ongedichte’.

```
theorem
  not ex n st for m holds n >= m
proof

  assume not thesis;
  then consider n such that
A1: for m holds n >= m;

  set n’ = n + 2;

  n’ > n by XREAL_1:31;

  then not for m holds n >= m;

  hence contradiction by A1;

end;
```

google wiskunde meisjes ↦ ⟨http://www.wiskundemeisjes.nl/⟩

and a more serious example: **a demo session in Spain**

Problem [B2 from IMO 1972]

f and g are real-valued functions defined on the real line. For all x and y,

$$f(x+y) + f(x-y) = 2f(x)g(y).$$

f is not identically zero and $|f(x)| \leq 1$ for all x. Prove that $|g(x)| \leq 1$ for all x.

formal proof sketch (Isabelle)

```
theorem IMO:
  assumes "ALL (x::real) y. f(x + y) + f(x - y) = (2::real) * f x * g y"
  and "~ (ALL x. f(x) = 0)" and "ALL x. abs(f x) <= 1"
  shows "ALL y. abs(g y) <= 1"
proof (clarify, rule leI, clarify)
  obtain k where "isLub UNIV {z. EX x. abs(f x) = z} k" sorry
  fix y assume "abs(g y) > 1"
  have "ALL x. abs(f x) <= k / abs(g y)"
  proof
    fix x
    have "2 * abs(g y) * abs(f x) = abs(f(x + y) + f(x - y))" sorry
    have "... <= abs(f(x + y)) + abs(f(x - y))" sorry
    have "... <= 2 * k" sorry
    show "abs(f x) <= k / abs(g y)" sorry
  qed
  hence "isUb UNIV {z. EX x. abs(f x) = z} (k / abs(g y))" sorry
  have "k / abs(g y) < k" sorry
  show False sorry
qed
```

fragment of the full formalization

```
proof (clarify, rule leI, clarify)
  obtain k where "isLub UNIV {z. EX x. abs(f x) = z} k"
    by (subgoal_tac "EX k. ?P k", force, insert prems,
        auto intro!: reals_complete isUbI setleI)
  hence a: "ALL x. abs(f x) <= k" by (intro allI, rule isLubD2, auto)
  fix y assume "abs(g y) > 1"
  have "ALL x. abs(f x) <= k / abs(g y)"
  proof
    fix x
    have "2 * abs(g y) * abs(f x) = abs(f(x + y) + f(x - y))"
      by (insert prems, auto simp add: abs_mult)
    also have "... <= abs(f(x + y)) + abs(f(x - y))"
      by (rule abs_triangle_ineq)
    also from a have "... <=  k + k" by (intro add_mono, auto)
    also have "... <= 2 * k" by auto
    finally show "abs(f x) <= k / abs(g y)"
      by (subst pos_le_divide_eq, insert prems,
          auto simp add: pos_le_divide_eq mult_commute)
```

etcetera

is formalization useful?

what does it buy me as a mathematician?

- *nothing*

  (you will tell the proofs to the computer, not the other way around)

- actually, it *does* buy you something:
  - your mathematics will be **utterly correct**
  - your mathematics will be **utterly explicit**

- humans are fallible
- computer programs always have bugs

how can we possibly promise **utter** correctness?

de Bruijn criterion

have a **very** small (part of the) program guarantee the correctness

HOL Light kernel: 542 lines = 17 pages

+ proof of correctness of HOL Light kernel has been formalized

(but: what if **definitions** are incorrect?)

how difficult is it?

de Bruijn factor

$$\frac{\text{size of formalization}}{\text{size of \LaTeX\ source of informal mathematics}} \approx 4$$

de Bruijn factor in time

$$\frac{\text{time to formalize}}{\text{time to understand the mathematics}}$$

is much larger

time to formalize one page from a textbook $\approx$ about one week

the state of the art: things that have been formalized

list of **100** nice theorems

1. The Irrationality of the Square Root of 2
2. Fundamental Theorem of Algebra
3. The Denumerability of the Rational Numbers
4. Pythagorean Theorem
5. Prime Number Theorem
6. Gödel's Incompleteness Theorem
7. Law of Quadratic Reciprocity
8. The Impossibility of Trisecting the Angle and Doubling the Cube
9. The Area of a Circle
10. Euler's Generalization of Fermat's Little Theorem

. . .

not formalized yet:

12. The Independence of the Parallel Postulate
13. Polyhedron Formula

. . .

formalized:	**77**
HOL Light	63
Coq	38
ProofPower	37
Mizar	35
Isabelle	33

google $\boxed{\text{100 theorems}}$ $\mapsto$ ⟨http://www.cs.ru.nl/~freek/100/⟩

serious theorems that have been formalized

- **first incompleteness theorem**
  nqthm, Natarajan Shankar
  Coq, Russell O'Connor
  HOL Light, John Harrison

- **fundamental theorem of algebra**
  Mizar, Robert Milewski
  HOL Light, John Harrison
  Coq, Herman Geuvers & others

- **Jordan curve theorem**
  HOL Light, Tom Hales
  Mizar, Artur Korniłowicz & others

- **prime number theorem**
  Isabelle, Jeremy Avigad

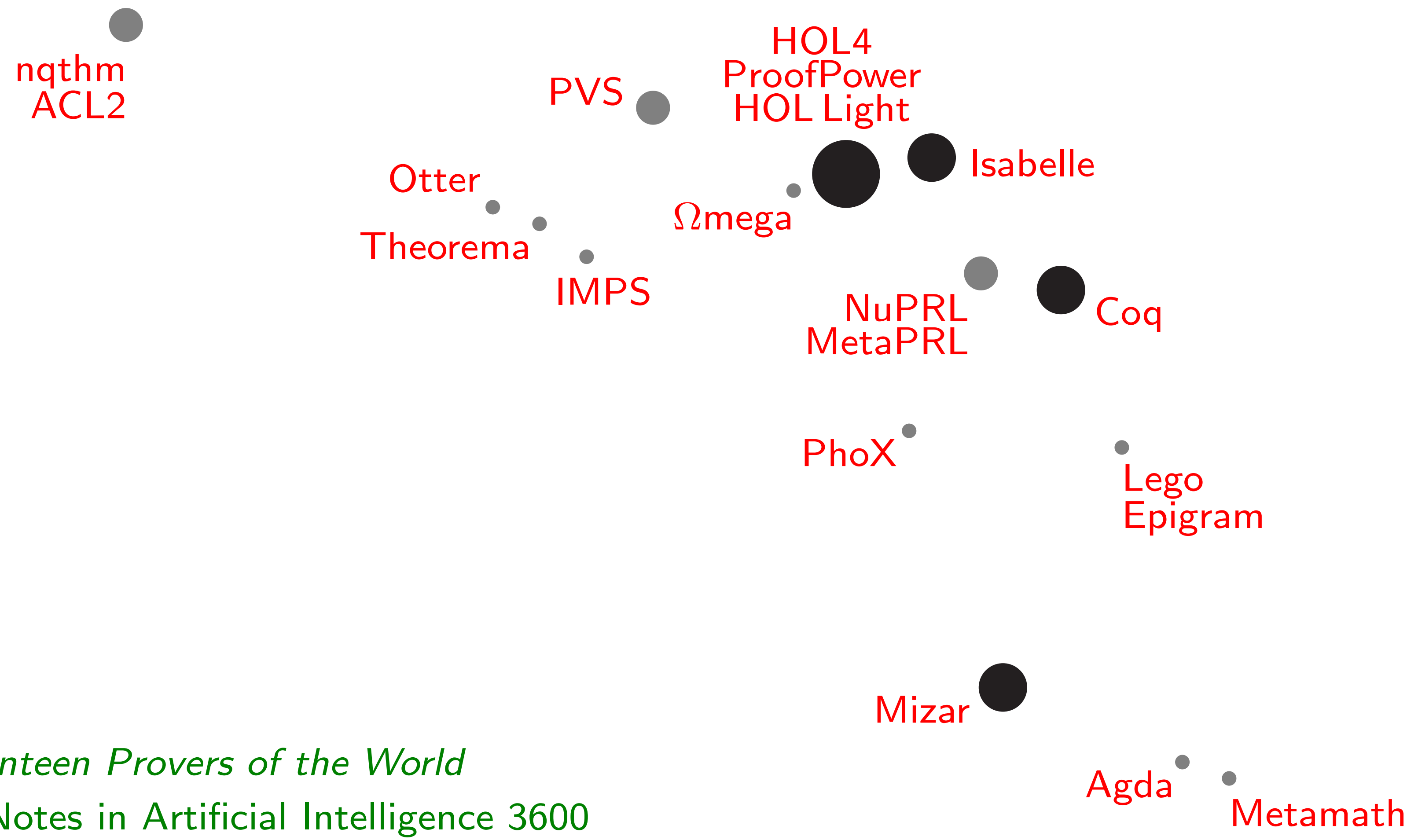
- **four color theorem**
  Coq, Georges Gonthier

0.03% of the four color theorem formalization

```
Lemma unavoidability : reducibility -> forall g, ~ minimal_counter_example g.
Proof.
move=> Hred g Hg; case: (posz_dscore Hg) => x Hx.
step Hgx: valid_hub x by split.
step := (Hg : pentagonal g) x; rewrite 7!leq_eqVlt leqNgt.
rewrite exclude5 ?exclude6 ?exclude7 ?exclude8 ?exclude9 ?exclude10 ?exclude11 //.
case/idP; apply: (@dscore_cap1 g 5) => x n Hn Hx Hgx// y.
pose x := inv_face2 y; pose n := arity x.
step ->: y = face (face x) by rewrite /x /inv_face2 !Enode.
rewrite (dbound1_eq (DruleFork (DruleForkValues n))) // leqz_nat.
case Hn: (negb (Pr58 n)); first by rewrite source_drules_range //.
step Hrp := no_fit_the_redpart Hred Hg.
apply: (check_dbound1P (Hrp the_quiz_tree) _ (exact_fitp_pcons_ Hg x)) => //.
rewrite -/n; move: n Hn; do 9 case=> //.
Qed.
```

the state of the art: the four best systems

proof assistants for mathematics

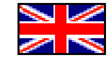
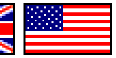


The Seventeen Provers of the World

Lecture Notes in Artificial Intelligence 3600

google $\boxed{\text{provers}}$ $\mapsto$ ⟨http://www.cs.ru.nl/~freek/comparison/⟩

first system: **HOL Light**

John Harrison, University of Cambridge $\leadsto$ Intel Corporation  

advantages	very elegant system
	strong automation
disadvantages	not really well suited for abstract algebra
	unreadable proof scripts

```
let LEMMA1 = prove
 (‘(!x y. f(x + y) + f(x - y) = &2 * f(x) * g(y)) /\ (!x. abs(f x) <= &1)
   ==> !l x. abs(f x * (g y) pow l) <= &1‘,
  DISCH_THEN(STRIP_ASSUME_TAC o GSYM) THEN INDUCT_TAC THEN
  ASM_SIMP_TAC[real_pow; REAL_MUL_RID] THEN GEN_TAC THEN MATCH_MP_TAC
   (REAL_ARITH ‘abs((&2 * a * b) * c) <= &2 ==> abs(a * b * c) <= &1‘) THEN
  ASM_SIMP_TAC[] THEN FIRST_ASSUM(MP_TAC o SPEC ‘x + y‘) THEN
  FIRST_ASSUM(MP_TAC o SPEC ‘x - y‘) THEN REAL_ARITH_TAC);;
```

second system: **Mizar**

Andrzej Trybulec, Białystok, Poland

advantages	readable proof scripts
	closest to actual mathematics
disadvantages	no first class binders (limits, sums, integrals)
	no user automation

- *procedural*

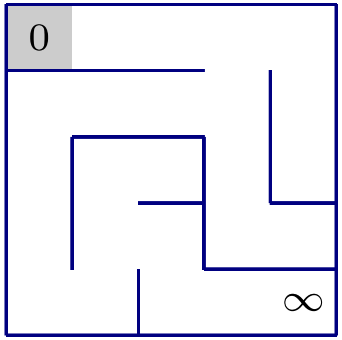
  HOL Light, Coq, Isabelle

  E E S E N E S S S W W W S E E E

- *declarative*

  Mizar, Isabelle

  (0,0) (1,0) (2,0) (3,0) (3,1) (2,1) (1,1) (0,1) (0,2) (0,3) (0,4) (1,4) (1,3) (2,3) (2,4) (3,4) (4,4)



third system: **Isabelle**

Larry Paulson, University of Cambridge 

Tobias Nipkow & Makarius Wenzel, Technical University Munich 

advantages	automation like HOL Light
	readable like Mizar
disadvantage	not really well suited for abstract algebra

- *set theory* (‘ZFC’)
- *type theory* $\rightsquigarrow$ each object has a ‘type’

  recursion/induction hardwired into the foundations
- *higher order logic* = weak set theory, also typed

  very simple and elegant

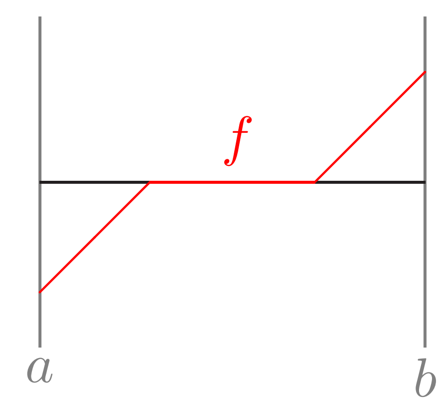
  not as expressive as set theory and type theory

fourth system: **Coq**

Gérard Huet & Thierry Coquand & many others, INRIA, Paris

advantages	automation like HOL Light and Isabelle expressive like Mizar
disadvantages	baroque foundations designed for intuitionistic mathematics

intermediate value theorem is intuitionistically not valid



google $\boxed{\text{intuitionism questions}}$ $\mapsto$ ⟨`http://www.intuitionism.org/`⟩

the state of the art: current projects

flyspeck

FlysPecK = Formal Proof of Kepler

Tom Hales' proof of Kepler's conjecture:
3 gigabytes of computer programs and data

referees did not understand it

- 'normal part' published in the *Annals of Mathematics*
- 'computer part' published in *Discrete and Computational Geometry*

2003: flyspeck project $\rightsquigarrow$ convincing the world
various prover communities involved: HOL Light, Coq, Isabelle

the three theorems everyone always starts talking about:

- ~~*four color theorem*~~

  Georges Gonthier, 2004

- *Fermat's last theorem*

  probably too big a hurdle yet . . .

- *classification of finite simple groups*

Georges Gonthier now has started work on the
odd order theorem = Feit-Thompson theorem

> *It takes a professional group theorist about a year of hard work to understand the proof completely* [. . .]
>
> — Wikipedia

outlook

two common misunderstandings

- **this will never be big: formalization is just too much work**

  misunderstanding: underestimating technology

  After formalizing the prime number theorem, I was struck with near certainty that, within a few decades, formally verified mathematics will become the norm. [...] *there are no major conceptual hurdles that need to be overcome; all it will take is clear thinking, sound engineering, and hard work.*

  — Jeremy Avigad

- **'I know mathematics, I can do this much better'**

  Paul Cohen, Harvey Friedman, Arnold Neumaier, *etcetera*

  misunderstanding: image of the computer as a research assistant

the best computer game in the world

formalization is like

- **programming**
  but no bugs, and not as trivial
- **doing mathematics**
  but completely transparent, and the computer helps

if you don't like one of them, you won't like formalization
if you like both, you will like formalization **very** much

> *Coq proofs are developed interactively* [...] *Building such scripts is surprisingly addictive, in a videogame kind of way* [...]
>
> — Xavier Leroy

- ancient greeks:

  proof

- end nineteenth century:

  rigor

- start twenty-first century:

  formalization of mathematics

will formalization become commonplace?

'killer app' for formalization has not yet been found . . .

current technology already very attractive:

- mathematics that is **utterly correct**
- mathematics that is **utterly explicit**

things will really become interesting when:

$$\textit{time needed for formalization} < 3 \cdot \textit{time needed for referee checking}$$