Theorem Provers Part 3: Why?

Thomas Sewell

Dec 2024

Theorem Provers Part 3: Why?

This lecture is about formal logic & mathematics as well its computerised version.

Why? What do we gain by doing our proofs in a formal style?

What does a software implementation add?

The Hilbert Program

David Hilbert (1862 - 1943) was an enormously influential mathematician.

(Holbert is a pun on HOL + Hilbert.)

Probably noone (human) will ever know all of the mathematics of their day. Hilbert might have come the closest.

In 1900, Hilbert presented a list of problems he hoped to be solved in the 20th century, setting the stage for other lists like the “millenium problems”.

Several of these relate to the “Hilbert program”.

A Potted History of Logic in Mathematics

- $\approx$ 300BC: Euclid's axioms and proofs.
  ldots
- 1800s: a huge explosion of mathematics, and various crises.
  - Calculus and the method of infinitesimals.
  - Limits and other infinite constructions.
- $\approx$ 1900:
  - Formal presentation of logic.
  - Set theory & Principia Mathematica.
  - Hilbert program: Develop an ideal logic.
- 1930: Gödel's incompleteness theorem.
- 1980s on: Formalisation of mathematics in ITPs.

The End of the Hilbert Program

The goal of the Hilbert program was to develop a good logic for the formalisation of all mathematics.

- Axiomatize everything in some formal logic.
- Ground complex and ambiguous systems (e.g. real analysis) in the simpler logic.
- Ensure consistency.
- Ensure decidability.

Gödel's results derail the program.

Good-Enough Solutions

Hilbert's hope of an ideal logic with "killer app" features like decidability is impossible.

Meanwhile, the 19th-century crises have ended, and many mathematicians feel that the Zermelo - Fraenkel set theory is good enough.

Good-Enough Solutions

Hilbert's hope of an ideal logic with "killer app" features like decidability is impossible.

Meanwhile, the 19th-century crises have ended, and many mathematicians feel that the Zermelo - Fraenkel set theory is good enough.

From the 1980s on, there has been some interest in formalising mathematics using a software tool, i.e. a theorem prover. Many of these tools can cover only a subset of mathematics, but this subset may be good enough, given the strong argument for correctness.

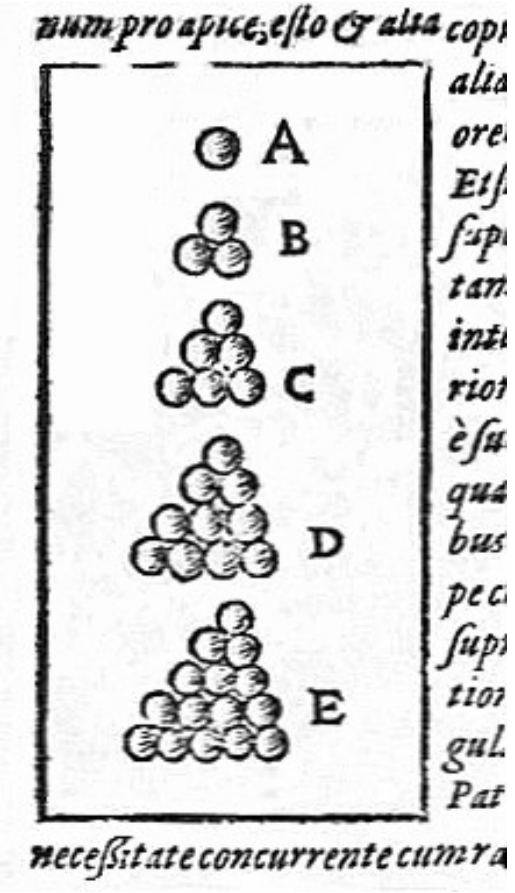
The Kepler Conjecture

The Kepler conjecture was stated in 1611.

The claim is that the ideal packing of spheres has the same density as the greengrocer or cannonball packing.

Gauss proved this for all regular arrangements in 1831.

It was Hilbert's 18th problem in 1900.



The Kepler Conjecture

In 1953, Tóth showed this could be reduced to a finite number of calculations, and observed that a computer could do this in principle.

In 1998, Thomas Hales and Samuel Ferguson announced they had completed the proof.

In 2003, the Annals of Mathematics agreed to publish the result, actually appearing in 2005.

The Kepler Conjecture

In 1953, Tóth showed this could be reduced to a finite number of calculations, and observed that a computer could do this in principle.

In 1998, Thomas Hales and Samuel Ferguson announced they had completed the proof.

In 2003, the Annals of Mathematics agreed to publish the result, actually appearing in 2005.

The proof included an exhaustive search done by a computer program.

The standard mathematical peer-review process didn't apply well.

The Flyspeck Project

From 2003-2014, Hales led the Flyspeck project.

The original proof was formalised and replayed using a combination of HOL-Light and Isabelle/HOL.

By its completion, the project had involved dozens of academics, and the final computation ran for CPU years on the Microsoft Azure cloud.

- `https://code.google.com/archive/p/flyspeck/wikis/AnnouncingCompletion.wiki`

21st Century Crisis?

The Kepler/Flyspeck proof is unusual, but also part of a trend.

Proofs are getting larger and harder to check.

In March 2023, Campos, Griffiths, Morris, and Sahasrabudhe published on arXiv a proof of an improvement to the upper bound on Ramsey numbers, generating a lot of discussion and excitement.

In November 2023, Bhavik Mehta published a formalisation of the work in the Lean theorem prover. This effectively settled the discussion about its correctness.

This has also generated a lot of excitement: probably for the first time, the computer formalisation has arrived in similar time to the community review process.

Software Verification

The other key application for theorem provers is in formalising proofs about computer programs.

(Cut to an old slide.)

Safety-Critical Software

Software is

- written in a source language

```
int
lorem (int ipsum) {
  int dolor, sit, amet;
  int incididunt = labore (dolor, MAGNA);
  exercitation (labore2 ());
  return 42;
}
```

Safety-Critical Software

Software is

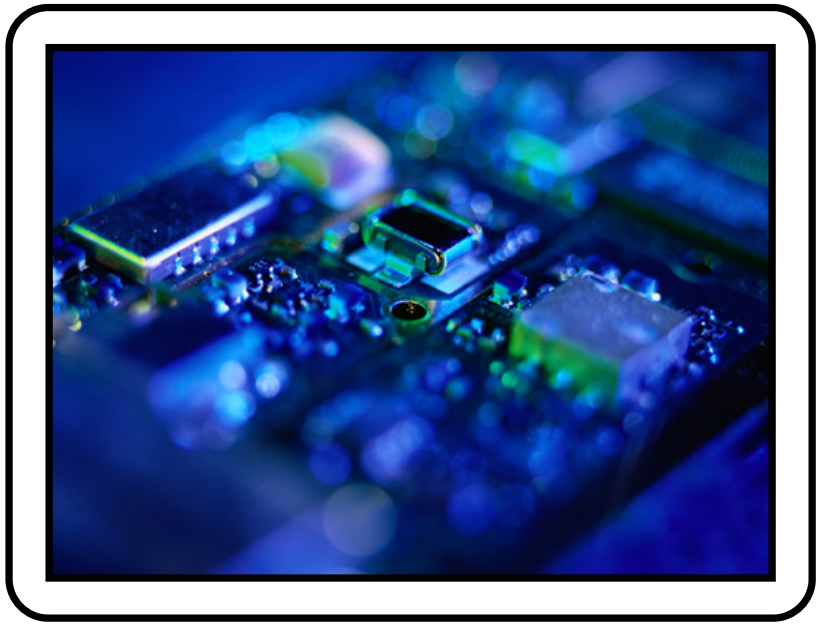
- written in a source language
- translated into a machine language

```
0000000c <lorem>:
   c:   e3510063       cmp     r1, #99 ; 0x63
  10:   e52d4004       push    {r4} ; (str r4, [sp, #-4]!)
  14:   ca000021       bgt     a0 <f+0x94>
  18:   e1a02181       lsl     r2, r1, #3
  1c:   e201c00f       and     ip, r1, #15
  20:   e2813001       add     r3, r1, #1
  24:   e2614063       rsb     r4, r1, #99     ; 0x63
  28:   e08cc002       add     ip, ip, r2
  2c:   e0801101       add     r1, r0, r1, lsl #2
  30:   e3530064       cmp     r3, #100        ; 0x64
  34:   e2044001       and     r4, r4, #1
  38:   e481c004       str     ip, [r1], #4
  3c:   e2820008       add     r0, r2, #8
  40:   0a000016       beq     a0 <f+0x94>
  44:   e3540000       cmp     r4, #0
  48:   0a000006       beq     68 <f+0x5c>
  4c:   e203200f       and     r2, r3, #15
```

Safety-Critical Software

Software is

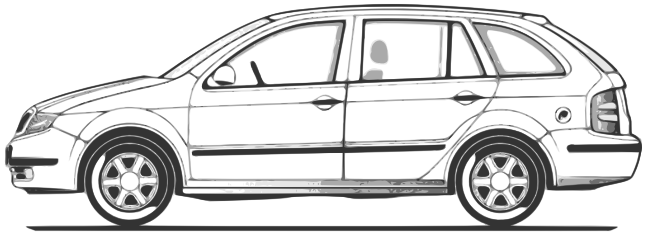
- written in a source language
- translated into a machine language
- loaded onto some silicon

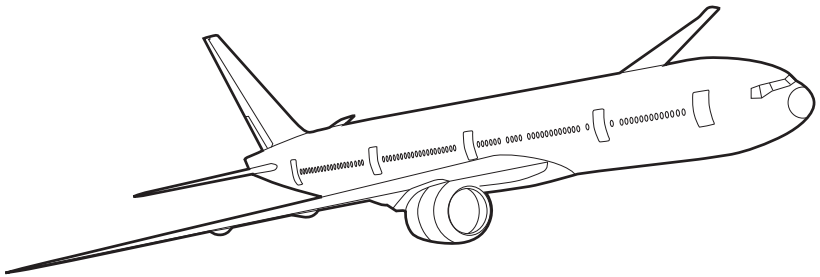


Safety-Critical Software

Software is

- written in a source language
- translated into a machine language
- loaded onto some silicon
- used in the real world.

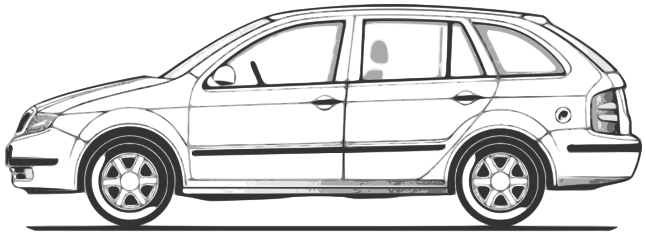


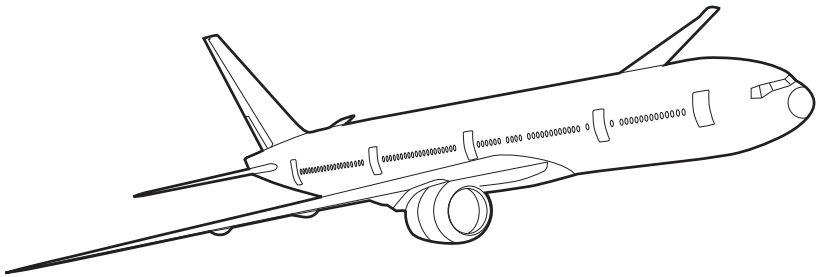


Safety-Critical Software

Software is

- written in a source language
- translated into a machine language
- loaded onto some silicon
- used in the real world.





Unlike our fellow engineers, software engineers don't have a good methodology for ensuring real-world safety, or avoiding expensive system failures.

Formal Methods

But we can prove things about computer programs!
(We saw how last week.)

This field of study is called “formal methods in software engineering”, or just “formal methods”.

Much like AI, formal methods has been around since the beginning of computing research. We both claim Alan Turing.

By the 1990s, formal methods had contributed many concepts and approaches, but was largely consigned to the “theory” part of the curriculum.

Software Verification

From the 2000s on, various projects proved the correctness of substantial pieces of essential software.

- CompCert (Coq), Leroy et al.
  - The C compiler is semantics-preserving from AST to assembly.
- Verisoft (Isabelle/HOL), Alkassar et al.
  - The custom C0 compiler, OS and application are all correct.
- seL4 (Isabelle/HOL), Klein et al.
  - The OS microkernel is safe and functionally correct.
- CakeML (HOL4), Kumar et al.
  - The functional compiler is correct, and compiles itself.
- Also CertiKOS, Iris, and many other projects.

Software Verification

From the 2000s on, various projects proved the correctness of substantial pieces of essential software.

- CompCert (Coq), Leroy et al.
  - The C compiler is semantics-preserving from AST to assembly.
- Verisoft (Isabelle/HOL), Alkassar et al.
  - The custom C0 compiler, OS and application are all correct.
- seL4 (Isabelle/HOL), Klein et al.
  - The OS microkernel is safe and functionally correct.
- CakeML (HOL4), Kumar et al.
  - The functional compiler is correct, and compiles itself.
- Also CertiKOS, Iris, and many other projects.
  - Your project here!

Proofs and Refutations

Imre Lakatos was a mathematician with a philosophical mindset.

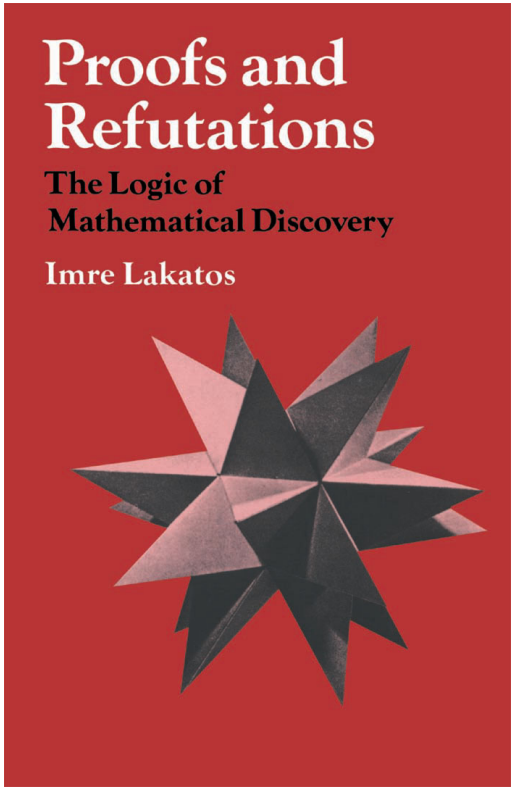
He wrote the definitive discussion of the philosophy of mathematics; in the sense that Popper & Kuhn set out to characterise the philosophy of science.

Proofs and Refutations

In my version of Lakatos' theory of mathematics:

- the gold standard may be formalisation
- dogmatic insistence on formalism is bad
- most mathematicians pose incomplete theories
- refutations and counterexamples are progress

The most cynical version is that this is all just a social process.

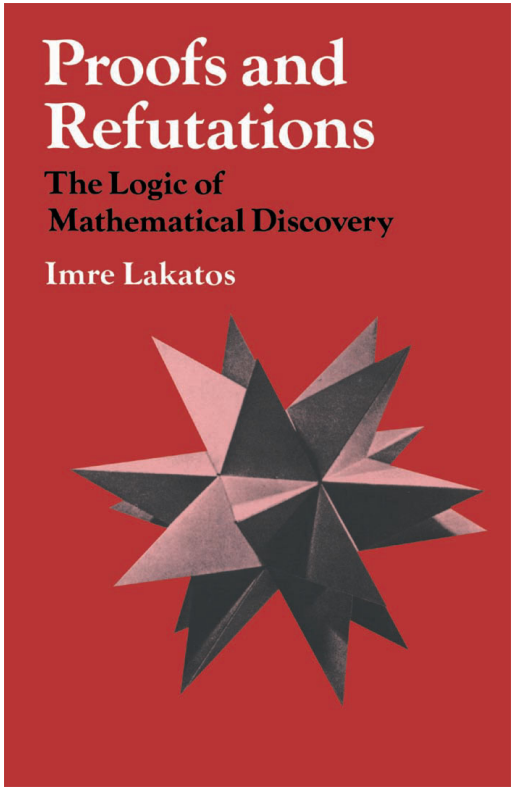


Proofs and Refutations

I would you read the book, and decide which interpretation you agree with.

- refutations and counterexamples are progress
- nothing is really proven or settled

Note that a machine-checked proof skewers one of the objections.



Recap

Recap. Why formalise logic, mathematics & computer science?

- To rescue what we can of the Hilbert program.
- To admit exhaustive proofs in mathematics.
- To reason about the correctness of critical software.
- To refute objections raised by readers of Lakatos.

Thanks for listening!

Questions?