Where the money is

Jeremy Avigad

Department of Philosophy
Department of Mathematical Sciences
Hoskinson Center for Formal Mathematics

Carnegie Mellon University

November 13, 2023

Formal methods

Formal methods are a body of logic-based methods used in computer science to

- write specifications for hardware, software, protocols, and so on, and
- verify that artifacts meet their specifications.

The same technology is useful for mathematics.

Formal methods in mathematics

Since the early twentieth century, we have known that mathematics can be represented in formal axiomatic systems.

Computational proof assistants allow us to write mathematical definitions, theorems, and proofs in such a way that they can be

- processed,
- verified,
- shared, and
- searched

by mechanical means.

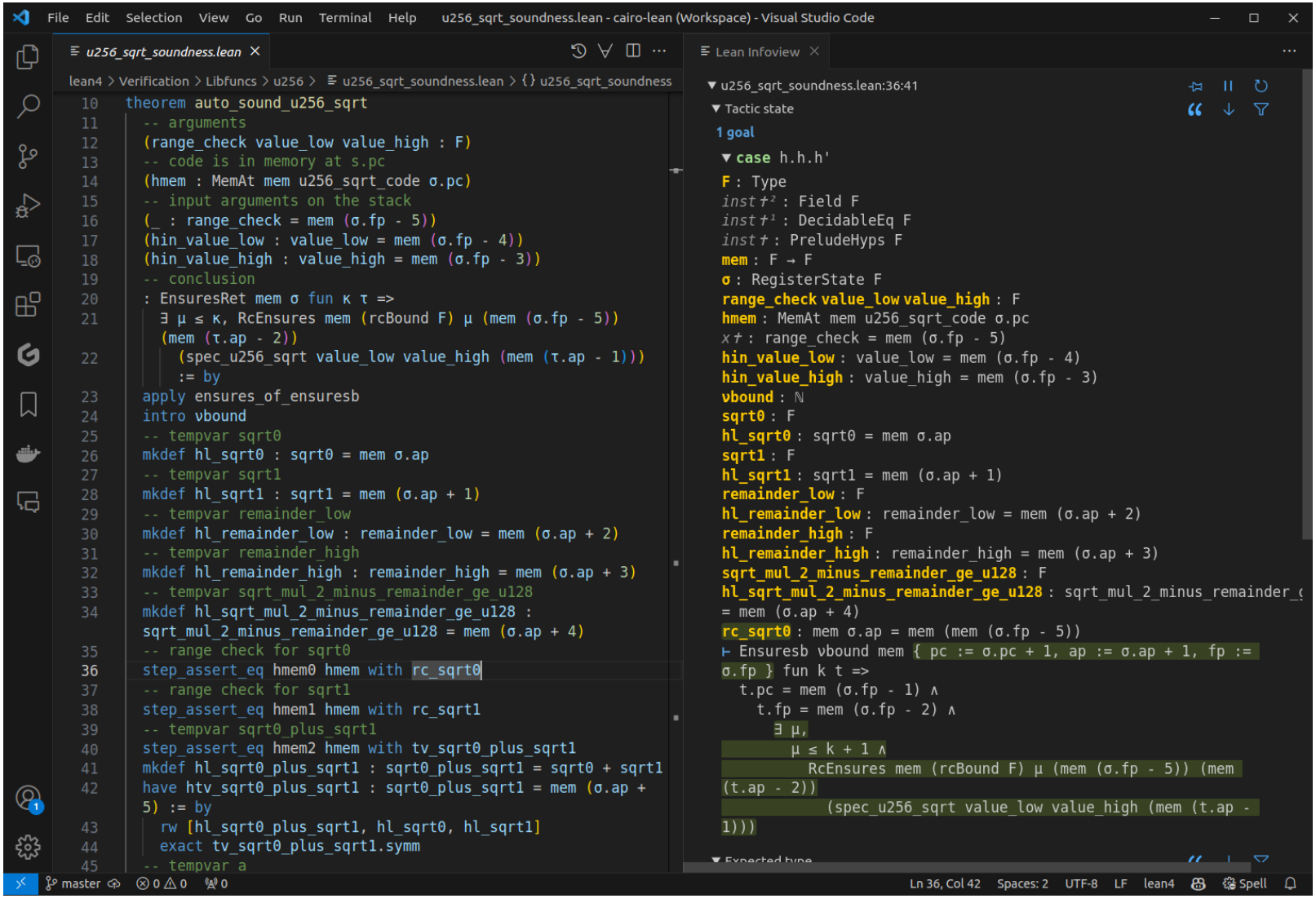
Formal methods in mathematics

The technology is useful:

- for verifying mathematics
- for building communal libaries
- for searching for mathematical results
- for collaborating
- as a gateway to AI
- as a gateway to automated reasoning tools
- for verifying mathematical and scientific software

I will focus on the last two.

Formal methods in mathematics



Where the money is

Formal verification is hard and time consuming.

As the story goes, when William Sutton was asked why he robbed banks, he replied, "because that's where the money is."

The only reason to verify certain pieces of mathematical software and certain mathematical results is because that's where the problems are.

Where the money is

We don't always need to verify everything down to axiomatic primitives.

Sometimes even just having a formal *specification* can be useful:

- It makes claims and models precise.
- It enables us to combine results from different sources.

We can then pick and choose which parts need to be verified.

Where the money is

I will discuss some examples where verification can make a difference.

- convex optimization
- applications of SAT solvers
- blockchain applications
- computational chemistry

Convex optimization

Numerical and symbolic methods for optimization are used extensively in engineering, industry, and finance.

Various methods are used to reduce problems to convex optimization problems.

1. Start with a problem you want to solve.
2. Reduce it to a convex optimization problem.
3. Transform it to the format required by a solver.
4. Solve it.
5. Check a certificate of correctness.
6. Report the results.

Convex optimization

Numerical and symbolic methods for optimization are used extensively in engineering, industry, and finance.

Various methods are used to reduce problems to convex optimization problems.

1. Start with a problem you want to solve.
2. Reduce it to a convex optimization problem.
3. Transform it to the format required by a solver.
4. Solve it.
5. Check a certificate of correctness.
6. Report the results.

Convex optimization

Sometimes sophisticated mathematical arguments are needed to reduce a problem. There are a variety of tricks and techniques to make problems convex.

Alexander Bentkamp, Ramon Fernández Mir, and I developed a prototype system in Lean:

- It provides a framework for defining problems and carrying out reductions.
- It has an extensible library of convex functions and their properties.
- It is situated within Lean's mathematical framework.
- It carries out "DCP transformations" automatically, and verifies them.

For now, we trust the floating point computations.

Applications of SAT solvers

Propositional satisfiability solvers are used in a variety of applications, including:

- hardware and software verification
- planning problems and AI
- cryptography
- proving combinatorial theorems.

Method:

- Start with the problem you want to solve.
- Reduce it to a SAT problem (or similar).
- Call the solver.
- Check the result (often an UNSAT proof).
- Report the result.

Applications of SAT solvers

Propositional satisfiability solvers are used in a variety of applications, including:

- hardware and software verification
- planning problems and AI
- cryptography
- proving combinatorial theorems.

Method:

- Start with the problem you want to solve.
- Reduce it to a SAT problem (or similar).
- Call the solver.
- Check the result (often an UNSAT proof).
- Report the result.

Applications of SAT solvers

Clever encodings and tricks are used to represent problems so that they can be solved efficiently.

Cayden Codel, James Gallicchio, Wojceich Nawrocki, Marijn Heule, and I are developing a Lean library to:

- carry out reductions and verify their correctness; and
- verify the results of SAT solvers and related tools.

Blockchain applications

Despite turmoil in decentralized finance, blockchain technology is here to stay. Billions of dollars are lost each year due to bugs in smart contracts.

StarkWare Industries provides a "layer two" solution:

- Write programs in a programming language, Cairo, such that successful termination guarantees a claim.
- Compile it to machine code.
- Encode the claim that the machine code runs to completion as the existence of solutions to a parametric family of polynomials.
- Publish on blockchain a short cryptographic certificate that guarantees the existence of the solutions.

Blockchain applications

Despite turmoil in decentralized finance, blockchain technology is here to stay. Billions of dollars are lost each year due to bugs in smart contracts.

StarkWare Industries provides a "layer two" solution:

- Write programs in a programming language, Cairo, such that successful termination guarantees a claim.
- Compile it to machine code.
- Encode the claim that the machine code runs to completion as the existence of solutions to a parametric family of polynomials.
- Publish on blockchain a short cryptographic certificate that guarantees the existence of the solutions.

Blockchain applications

With colleagues at StarkWare, we have:

- verified the encoding of termination claims as polynomials;
- written a proof-producing compiler from Cairo to machine code;
- verified components of the library, such as elliptic curve signature validation, down to the cryptographic certificates.

At this stage, we trust the cryptographic protocol, though others have worked on verifying those too.

Computational chemistry

I have learned from Tyler Josephson that Monte Carlo methods are fundamental to computational chemistry. Thousands of papers on simulations are published each year.

Method:

- Start with a model of a physical system and a quantity you want to compute.
- Design a Monte Carlo algorithm to compute it.
- Implement it, and run it.
- Report the results.

Computational chemistry

I have learned from Tyler Josephson that Monte Carlo methods are fundamental to computational chemistry. Thousands of papers on simulations are published each year.

Method:

- Start with a model of a physical system and a quantity you want to compute.
- Design a Monte Carlo algorithm to compute it.
- Implement it, and run it.
- Report the results.

Computational chemistry

Researchers use tricks and heuristics to make a Markov chain Monte Carlo (MCMC) algorithm converge quickly.

A *detailed balance* condition ensures that the samples converge to the correct distribution. Violations lead to discrepancies in the literature.

Tyler and I proposed to:

- Verify detailed balance conditions in Lean.
- Implement and verify probabilistic programs to sample from the relevant distributions.

Conclusions

Morals:

- Talk to people outside formal methods. There are important verification problems everywhere.
- Be flexible. Choose your battles.
- Be creative.
- Be proud. Verification matters.