AI for Mathematicians

Jeremy Avigad

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

Carnegie Mellon University

December 14, 2024

AI for Mathematics

Two traditions in AI:

- symbolic AI and formal methods
- machine learning and neural networks.

The strengths are complementary:

- Symbolic AI is good at getting details right but gets lost.
- Machine learning is good at synthesizing data but we don't know what the results mean.

Both are important for mathematics.

Formal Methods

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

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

The same technology is useful for mathematics.

I use "formal methods in mathematics" and "symbolic AI for mathematics" roughly interchangeably.

Formal Methods

They are based on:

- formal languages
- formal proof systems
- formal semantics.

Complementary approaches:

- automated reasoning
- interactive theorem proving.

Formal Methods

A brief history of automated reasoning:

- late 19th century–early 20th century: birth of modern logic
- 1910's–1930's: decision procedures for fragments of first-order logic, linear arithmetic, linear integer arithmetic, real closed fields, algebraically closed fields
- 1929: The completeness theorem for first-order logic (Gödel)
- 1950's–1960's: provers for propositional logic, first-order logic, Presburger arithmetic, geometry.

Formal Methods

Interactive theorem began later:

- late 1960's–1970's: Automath, Mizar (mathematics)
- 1970's:
  - the LCF framework (interaction, tactics)
  - HOL (hardware and software verification)
  - NQTHM (a precursor to ACL2)
- 1980's:
  - Isabelle
  - Coq
- 1990's: HOL Light

Lean

History:

- Launched by Leonardo de Moura in 2013.
- Mathlib launched in 2017.
- Lean 4 released in 2022.

Features:

- based on an expressive dependent type theory
- designed for both mathematics and software verification
- computational core (it's also a programming language)
- good support for classical reasoning
- good support for a large network of algebraic structures
- lively, energetic, enthusiastic, and supportive community.

Lean

Successes:

- Mathlib has 80K definitions, 180K theorems, 1.6M lines of code.
- The Lean Zulip channel has 10K subscribers, 800 active in any two-week period.
- Notable formalization projects include Liquid Tensor Experiment, improved upper bounds on Ramsey's theorem, the Sphere Eversion Project, polynomial Freiman–Ruzsa conjecture, a generalization of Carleson's theorem.
- It's a powerful platform for machine learning, e.g. DeepMind's AlphaProof.

AI for Mathematics

Machine learning support for interactive theorem proving:

- premise selection
- search
- copilots

ITP support for AI:

- Formal proofs provide a clear measure of success.
- Formal proofs are like computer code.
- Formal libraries provide good data.
- Interactive provers provide a clear signal e.g. for reinforcement learning.
- People are excited about formal proofs.

Outline

Contents:

- Formal methods for mathematics
- Formal methods and AI for mathematics
- Why AI for mathematics is important
- What is important for mathematics

Why AI for Mathematics is Important

Concerns about AI:

- How do we know the results are correct?
- Can we trust the results?
- Do we know what the results mean?
- Can we get explanations?
- Can we get justifications?

Why AI for Mathematics is Important

Mathematics provides a precise language with which we can:

- express ourselves
- write specifications
- demand justifications
- demand explanations.

It's a fundamental part of

- how we make sense of the world
- how we communicate with one another
- how we make decisions
- how we design and build things.

Why AI for Mathematics is Important

For AI to be useful to us, it has to be part of that process.

We want to

- ask questions
- understand the answers
- ask for explanations
- reason and deliberate
- explore ideas.

Why AI for Mathematics is Important

AI for mathematics is not just about getting the right answer.

It's about providing tools that help us reason mathematically.

AI for mathematics *is* AI for mathematicians.

And we are all mathematicians.

Benchmarks

Research in machine learning and automated reasoning is driven by benchmarks.

- They provide uniform and reproducible assessment.
- They provide clear measures of progress.

But benchmarks are often a poor proxy for the things we really care about.

Benchmarks

We want AI to be useful.

- Tools need to be easy to set up, install, and use.
- Tools need to integrate into existing workflows.
- We want help in specific contexts.

Challenges:

- Benchmarks are often drawn from a finished library.
- Benchmarks favor stand-alone problems.
- Reports of success are hard to interpret.
- Tools are often overtuned to the benchmarks.

The only way to find out what is useful is to get the mathematical community involved.

Example: miniCTX

When a mathematician wants to formalize something, they create a *project*, with Mathlib as a dependency.

Definitions and theorems are spread across multiple files.

Jiewen Hu, Thomas Zhu, and Sean Welleck, miniCTX: Neural Theorem Proving with (Long-)Contexts, provides a benchmark to evaluate theorem provers in such contexts.

It also explores methods for and the effects of including in-file and cross-file contexts.

Example: ImProver

Riyaz Ahuja, Jeremy Avigad, Prasad Tetali, and Sean Welleck, ImProver: Agent-Based Automated Proof Optimization explores the use of LLMs to *improve* proofs wrt different metrics.

Initial experiments focus on three metrics:

- length
- declarativity
- a mixture of the two.

Applications:

- user-end tools
- data augmentation
- optimizing data for training.

Example: ImProver

We evaluate the effects of various components:

- "chain of state" training and prompting
- refinement / best-of-n sampling techniques
- error correction
- MMR-based retrieval augmented generation from Mathlib and Theorem Proving in Lean.

Shortcomings:

- The metrics are simplistic.
- The process is slow.
- The process is expensive.

Summarizing the Examples

The experiments are aimed at developing useful tools, but more work is needed to:

- get tools in the hands of users
- get the cost down
- make the tools faster
- tune the tools to achieve what users really want.

Getting Priorities Straight

A common distinction:

- *Basic science:* understanding the basic principles with which we can train a system to exhibit intelligent behavior.
- *Engineering:* meeting the design challenges that are needed to deploy such systems and make them useful.

I want to push back against this way of looking at things.

Artificial Intelligence

Intelligent beings:

- function in real time
- interact with their environment
- are sensitive to and adapt to context
- make do with available resources.

Lab benchmarks are only proxies for real-world intelligence.

Highly recommended:

Jason Rute, The Last Mile: How do we make AI theorem provers which work in the real world for real users and not just on benchmarks?

Artificial Intelligence

"Engineering" goals:

- usability
- transparency
- predictability
- speed
- limited compute requirements
- limited cost requirements
- flexibility
- adaptability
- robustness.

"Intelligence" requires scoring well on these axes.

Parting Claims

Getting AI to do mathematics well requires a combination of machine learning and formal methods.

> *I may be wrong; this is an empirical question.*

Getting AI to do mathematics well includes having it generate formally checkable proofs.

> *Why not? I am a mathematician, and I want formally checkable proofs.*

Getting AI to do mathematics well means making it useful for mathematicians.

> *This is what we mean by "doing mathematics well."*

And we are all mathematicians.

Summary

AI for mathematics

=

AI for mathematicians

=

AI for everyone