Machine learning and symbolic AI for mathematics

Jeremy Avigad

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

Carnegie Mellon University

December 11, 2023

Outline

- Machine learning and symbolic AI
- Machine learning for mathematics
- Formal methods in mathematics
- Automated reasoning for mathematics
- Automated reasoning in Lean
- Keeping users in mind
- Machine learning and automated reasoning
- Machine learning and formal methods
- Machine learning tools for Lean

Symbolic AI and machine learning

Symbolic AI:

- logic-based representations
- precise, exact
- explicit rules of inference

Machine learning:

- distributed representations
- probabilistic, approximate
- based on lots of data

Symbolic AI and machine learning

Mathematics needs both.

- Machine learning can synthesize data and discover patterns and connections.
- Symbolic methods provides justification and explanation.

Synthesizing the two approaches is a key challenge:

- ML has intuitions but gets details wrong.
- Symbolic AI is precise but gets lost.

Moreover:

- We don't know and can't control what ML is doing.
- Symbolic AI can't solve problems we care about.

Symbolic AI and machine learning

Precise symbolic representations are central to human communication, collaboration, deliberation, and reasoning.

We need these to mediate our interactions with ML.

Understanding how to get the best of both worlds is a central challenge for AI.

Mathematical reasoning is a great place to start.

Symbolic AI and machine learning

"We may hope that machines will eventually compete with men in all purely intellectual fields. But which are the best ones to start with? Even this is a difficult decision. Many people think that a very abstract activity, like the playing of chess, would be best. It can also be maintained that it is best to provide the machine with the best sense organs that money can buy, and then teach it to understand and speak English. This process could follow the normal teaching of a child. Things would be pointed out and named, etc. Again I do not know what the right answer is, but I think both approaches should be tried."

Alan Turing, "Computing Machinery and Intelligence," 1950

Machine learning for mathematics

Some applications of ML to mathematics use ML to:

- search for interesting mathematical objects
- detect patterns in computational data.

Examples:

- Knot invariants (sensitivity analysis): Lackenby and Juhász with Davies and DeepMind.
- Kazhdan-Lustig polynomials and Bruhat intervals (sensitivity analysis): Williamson with Davies and DeepMind.
- Counterexamples in graph theory (reinforcement learning): Wagner
- Ribbons in knot theory (reinforcement learning): Gukov, Halverson, Manolescu, and Ruehle

Machine learning for mathematics

Some of the methods yield insights and intuitions, while others objects with verifiable properties.

The applications require symbolic representations (to compute the data, or carry out the search), but they do not require formal logic.

I won't talk about these applications.

Formal methods in mathematics

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.

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

Formal methods in mathematics

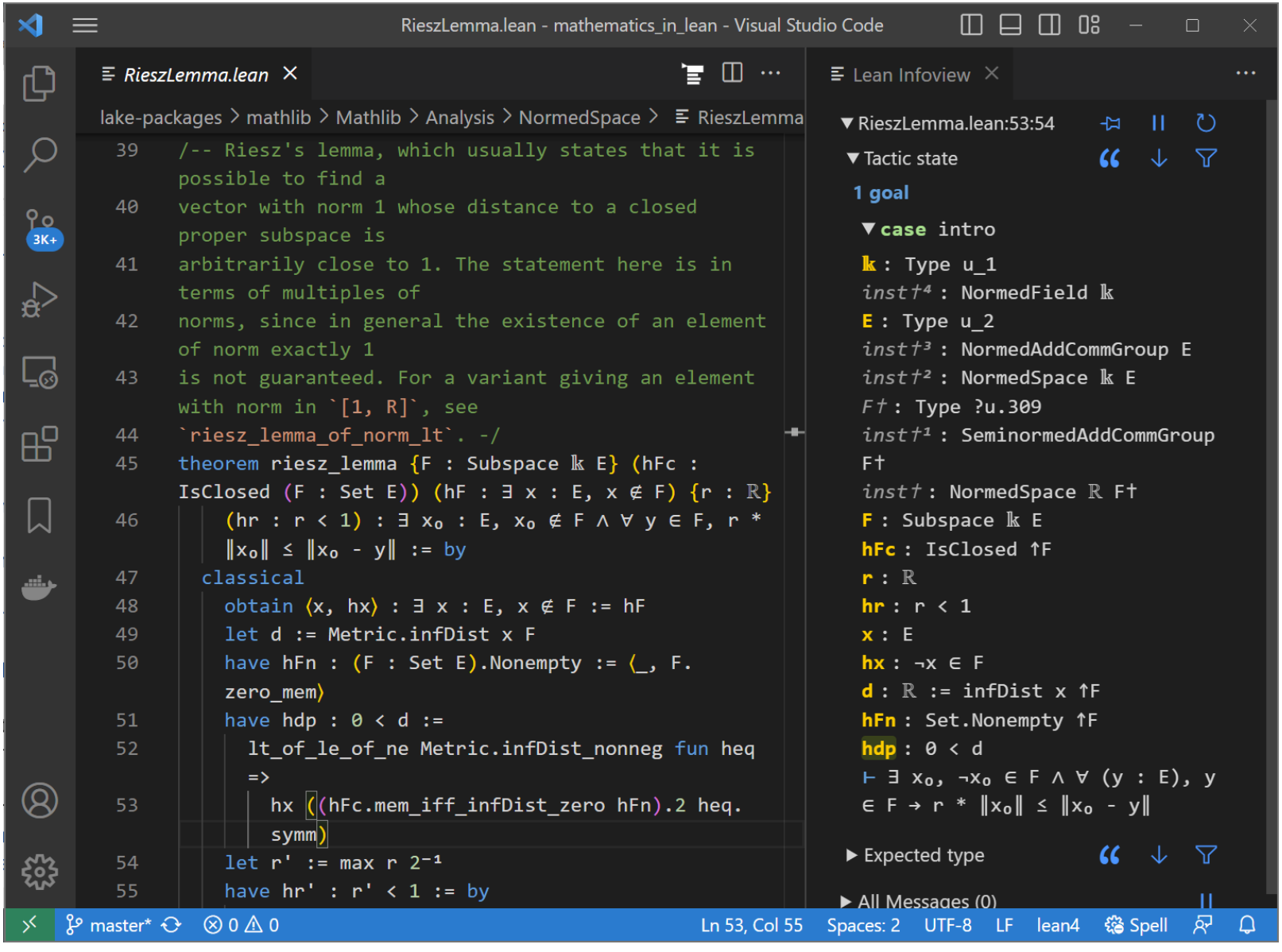
Since the early twentieth century, we have known that mathematics can be represented in formal axiomatic systems.

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



Formal methods in mathematics

The digitization of mathematical knowledge brings benefits:

- verification
- formal libraries
- search
- collaboration
- teaching
- use of automated reasoning
- interaction with machine learning

Question: how should machine learning interact with formal methods?

- formal methods can help machine learning
- machine learning can help formal methods

Automated reasoning for mathematics

With automated reasoning, one can distinguish between:

- domain-general methods vs. domain-specific methods
- search procedures vs. decision procedures

Domain-general: equational reasoning, propositional logic, first-order logic, combination methods

Domain-specific: linear integer arithmetic, linear real arithmetic, nonlinear equalities and inequalities, Gröbner bases, ...

ML seems most promising for domain-general search, where symbolic AI struggles the most.

Automated reasoning for mathematics

With automated reasoning, one can also distinguish between:

- Verification: checking the correctness of a theorem that we already have good reason to believe is true.
- Discovery: finding new theorems.

The line between these is not sharp.

Automated reasoning in proof assistants (Isabelle, HOL Light, Coq, Lean) provides examples of the first.

There are only isolated successes in the second domain. Many involve SAT solvers; see Marijn Heule's talk.

Automated reasoning for mathematics

Central tools:

- first-order theorem provers (Vampire, E, Zipperposition, ...)
- SMT solvers (Z3, CVC5, ...)
- SAT solvers (CaDiCaL, Kissat, ...)

Question: how can machine learning interact with automated reasoning?

- Machine learning can help automated reasoning.
- Automated reasoning can help machine learning.

Automated reasoning for Lean

Isabelle still has the best domain-general automation.

Domain-general automation for Lean:

- The simplifier, `simp`
- Aesop (Limperg, From)
- Duper (Clune, Qian, Bentkamp, in progress)
- The Lean-SAT library (Codel, Gallicchio, Nawrocki, in progress)
- Lean-Auto (Qian)
- `exact?`, `apply?`, `rw? rw-search`
- tab completion

It is remarkable how far Mathlib has gotten without a lot of automation.

Automated reasoning for Lean

How to prove a theorem using a SAT solver:

1. Reduce the statement of interest to the satisfiability or unsatisfiability of a propositional formula.
2. Send it to a SAT solver.

Cayden Codel, James Gallicchio, and Wojciech Nawrocki (with Marijn and me) are working on LeanSAT, which are libraries to:

1. Construct suitable encodings.
2. Verify the correctness of the reduction.
3. Verify the solver's results.

Goal: to have useful infrastructure for using SAT solvers reliably.

Automated reasoning for Lean

Josh Clune, Yicheng Qian, and Alex Bentkamp are working on Duper, a proof-producing superposition theorem prover for Lean.

It can be used as stand-alone automation, but also for proof reconstruction for a sledgehammer (more later).

Dependent type theory poses novel challenges.

Automated reasoning for Lean

Sending Lean proof goals to external tools requires preprocessing:

- The target logics are less expressive.
- Reasoning about mathematical structures is expensive.

Yicheng Qian is working on Lean-Auto.

- It exports problems to external first-order provers.
- It exports problems to SMT solvers.
- It provides essential preprocessing for Duper.

He is also interested in proof reconstruction.

Automated reasoning for Lean

In short, domain-general automation for Lean is now becoming available.

Questions:

- How can machine learning help?
- How can machine learning benefit?

Keeping the user in mind

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

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

But machine learning for mathematics should provide tools that are useful for mathematics.

Benchmarks are often a poor measure of that.

Keeping the user in mind

Shortcomings:

- Tools need to be easy to set up, install, and use.
- Tools need to integrate into existing workflows.
- We want help in specific contexts.
- Benchmarks are often drawn from a finished library.
- It's hard to tell whether successes are on the problems for which we really want help.
- Tools are overtuned to the benchmarks.

Mathematicians will find creative ways to use automated reasoning.

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

Keeping the user in mind

Successes of machine learning for mathematics:

- Ambitious mathematicians can learn how to use PyTorch, Colab, and so on.
- Isabelle's sledgehammer has used machine learning for premise selection for years.
- Terry Tao got a bit of help from ChatGPT when learning to use Lean.
- Github Copilot is mildly helpful when working with Lean.

We need to do better!

If we are not sensitive to what mathematicians want now, it is not clear whether we will make long-term progress.

Machine learning for automated reasoning

Ways that machine learning can support symbolic automated reasoning:

- premise selection
- instantiating universal / existential quantifiers
- proof sketches (auxillary claims and lemmas)
- guiding heuristics (clause selection, literal selection)

There are senses in which these are the only hard parts of automated reasoning.

Machine learning for automated reasoning

Notes:

- With the right premises and definitions, a complete first-order search is guaranteed to proceed.
- With the right Herbrand instantiations, validity is decidable (propositional logic plus ground equational reasoning).
- Introducing cuts can result in superexponential speedup.
- SAT solvers do well even though SAT is NP-complete.

So:

- Premise selection is important.
- Instantiating universal / existential quantifiers is important.
- Structured proofs and theories are important.
- Heuristics are important.

Machine learning and automated reasoning

How a sledgehammer works:

- Start with a goal in a proof assistant.
- Choose 200–300 theorems that might be helpful from a library of tens of thousands.
- Call a powerful, external theorem prover.
- If it succeeds, it will likely only use a handful of the background theorems.
- Use less powerful, proof-producing automation to construct a formal proof.

See Desharnais et al., "Seventeen provers under the hammer" for a recent assessment of the state of the art.

Machine learning and automated reasoning

Prospects for a sledgehammer for Lean:

- Thanks to Lean-Auto, we can send goals to an external prover and interpret the results.
- Thanks to Duper, we can reconstruct proofs.
- We still need to handle premise selection.

There is a substantial literature on using machine learning for premise selection. (See Mikula et al., "Magnushammer: A Transformer-based Approach to Premise Selection.")

Machine learning and automated reasoning

Premise selection for Lean:

- Geesing, "Premise selection for Lean 4" (MS thesis)
- Piotrowski, Mir, and Ayers, "Machine-Learned Premise Selection for Lean"
- Yang et al., "LeanDojo: Theorem Proving with Retrieval-Augmented Language Models"

We still need to adapt them for a sledgehammer and evaluate.

Machine learning and automated reasoning

Can machine learning help in other ways?

"Draft, Sketch, and Prove" (Jiang et al.) breaks a task into smaller pieces and suggests key terms.

Question: can it be made to work *in situ*?

Question: can machine learning guide the search itself?

That seems hard: symbolic methods carries out inferences much more quickly.

See Lample et al., "HyperTree Proof Search for Neural Theorem Proving."

Machine learning and automated reasoning

Can machine learning help SAT solvers?

See Marijn's talk tomorrow.

With Evan Lohn, we have had success in using machine learning to predict runtimes in determinations of unsatisfiability.

Machine learning and formal methods

Other ways machine learning can be useful:

- tutors
- code pilots
- premise selection for interactive proving
- semantic search (for both formal and informal content)
- autoformalization (of definitions, statements, proofs, theories)
- informalization
- explanations of proofs found by automation

With humans in the loop, there is a lower bar. ML needs only get close, and be helpful.

Can formal systems provide feedback and guidance for machine learning?

Machine learning tools for Lean

Here are some existing machine learning tools for Lean:

- llmstep (Welleck; suggests tactic steps)
- LeanCopilot (Song, Yang, and Anandkumar; suggests tactic steps)
- Moogle (semantic search)
- Morph labs Lean assistant
- Lean premise selection (Piotrowski, Fernández Mir, and Ayers)
- Github copilot
- ChatGPT

Machine learning tools for Lean

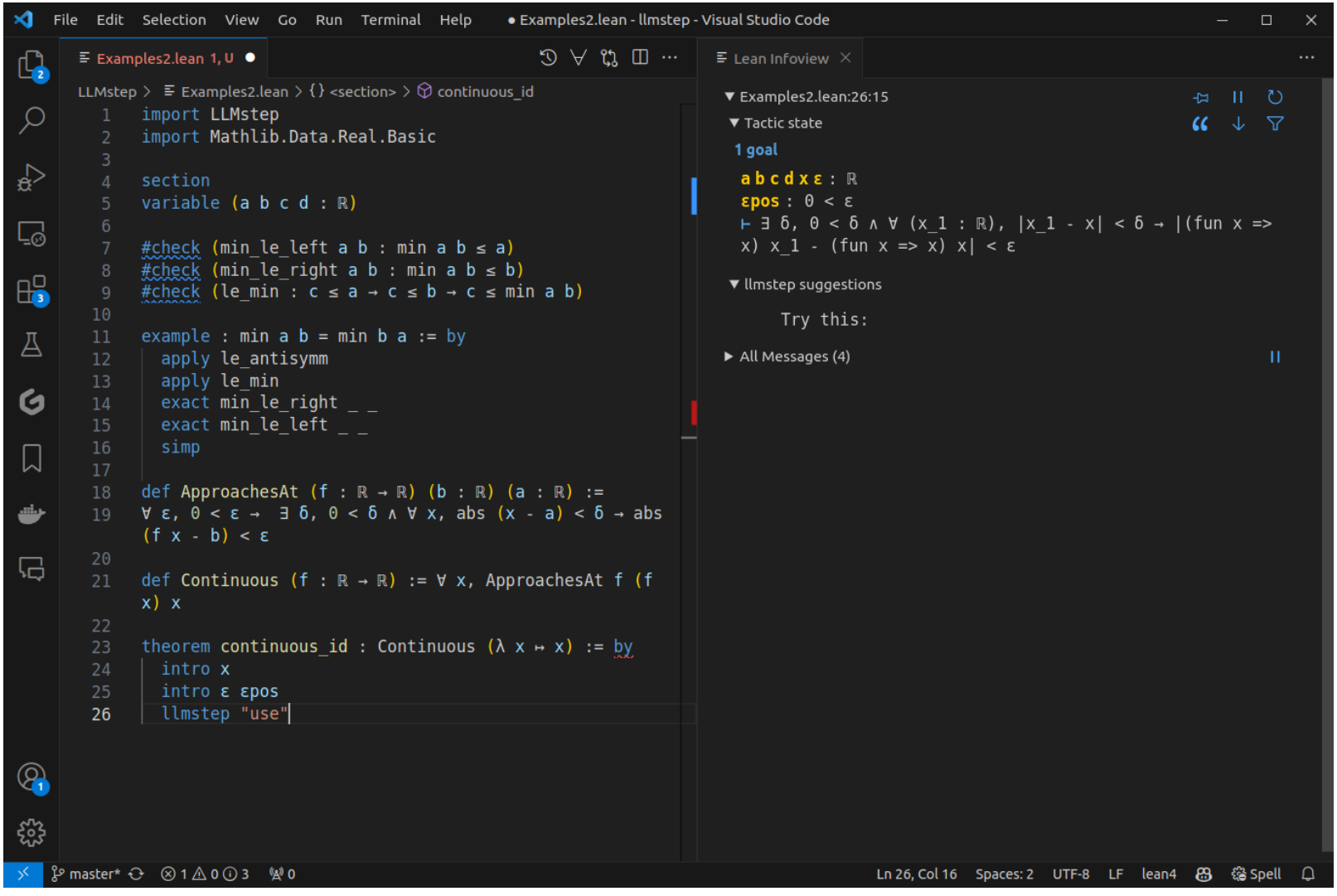
Practical challenges:

- Dependencies sometimes cause problems (Python, NodeJS).
- Platforms sometimes cause problems (Windows, Linux, MacOS).
- Fetching data files (wget, curl) can be slow.
- Configurations sometimes need tweaking.
- There are instabilities: server crashes, editor crashes, system crashes, etc.
- Cloud tools sometimes require registration and a key.
- Data is sometimes made public.

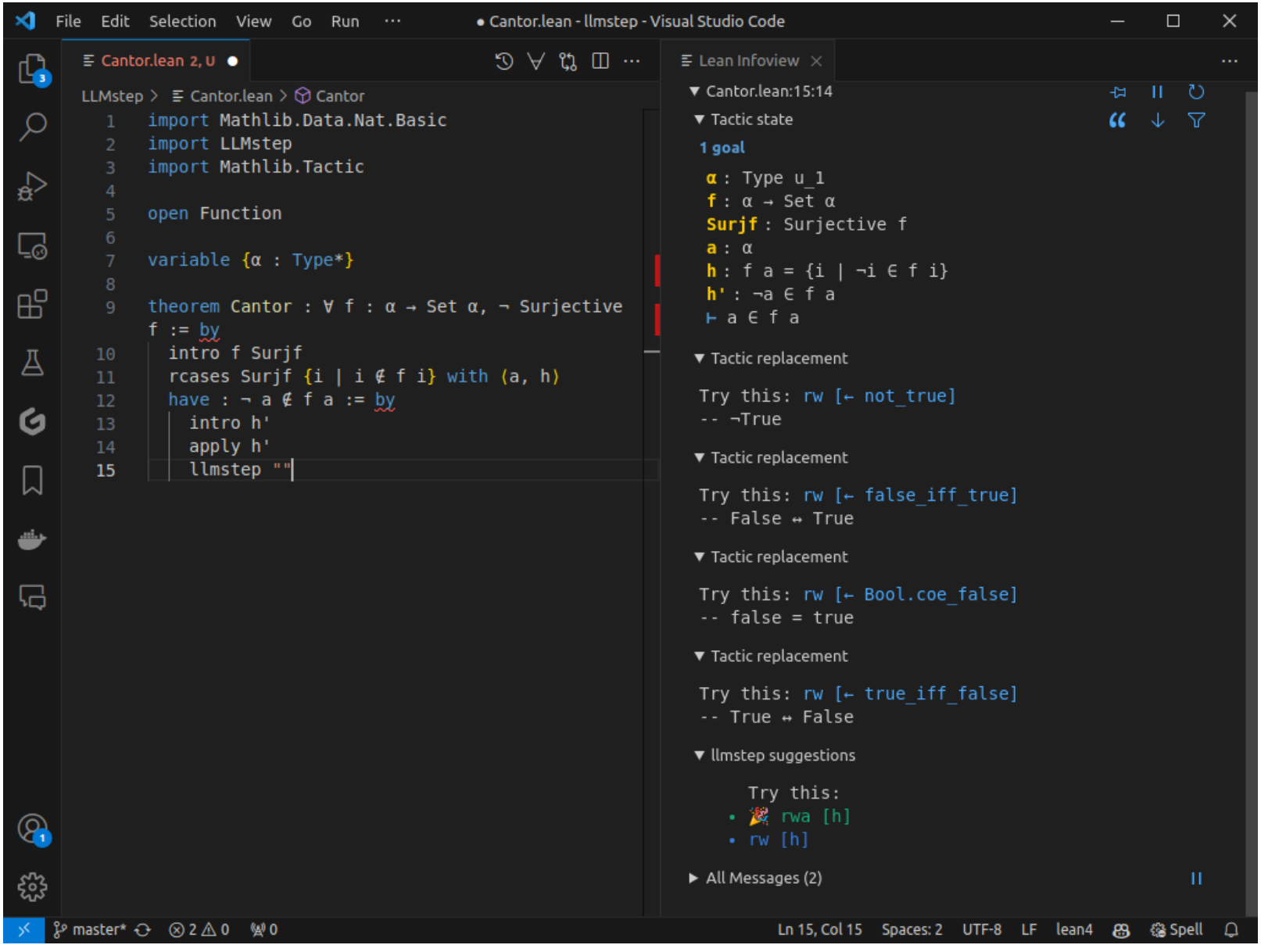
Machine learning tools for Lean

Some experiments with llmstep. . .

Machine learning tools for Lean



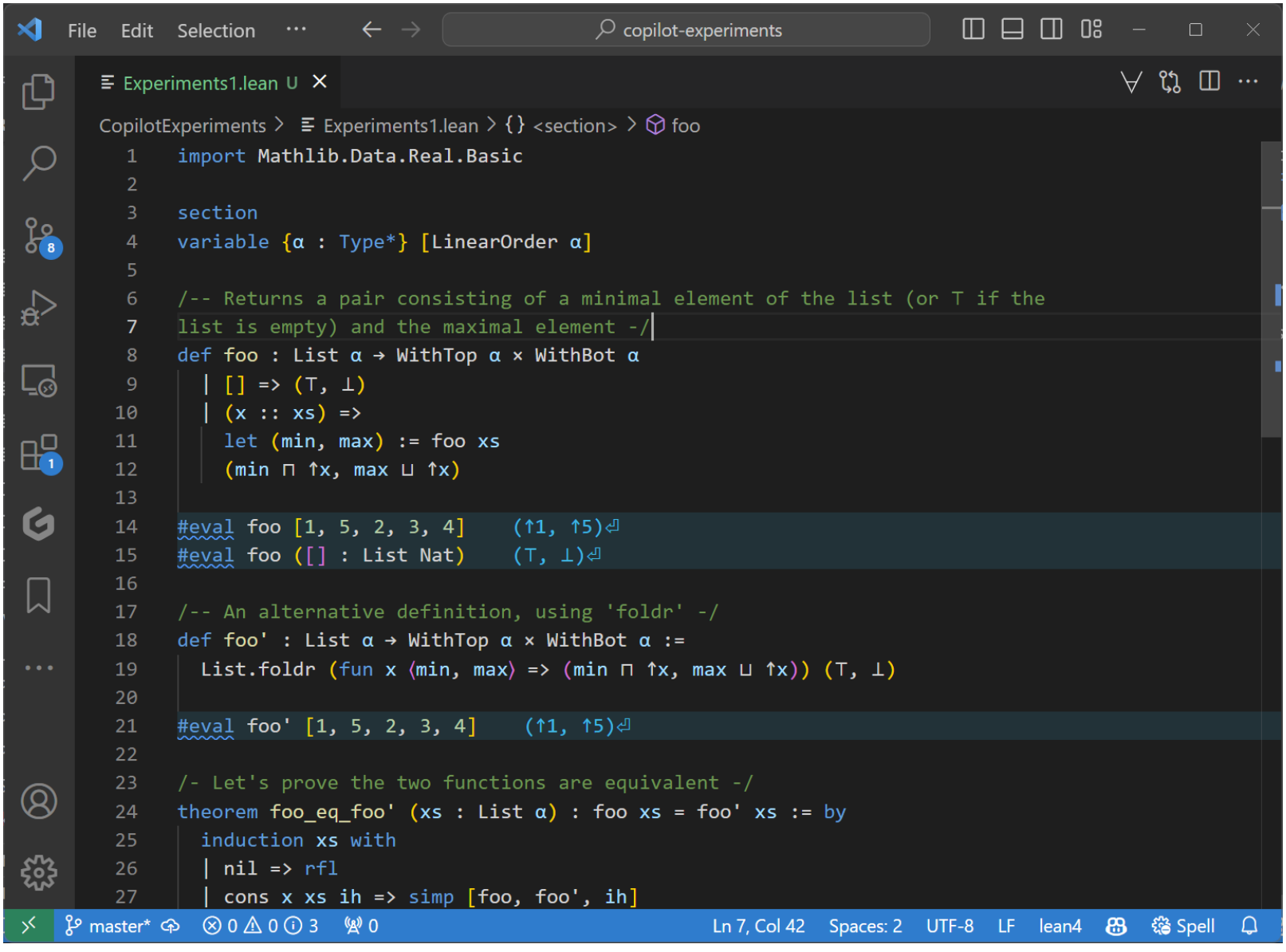
Machine learning tools for Lean



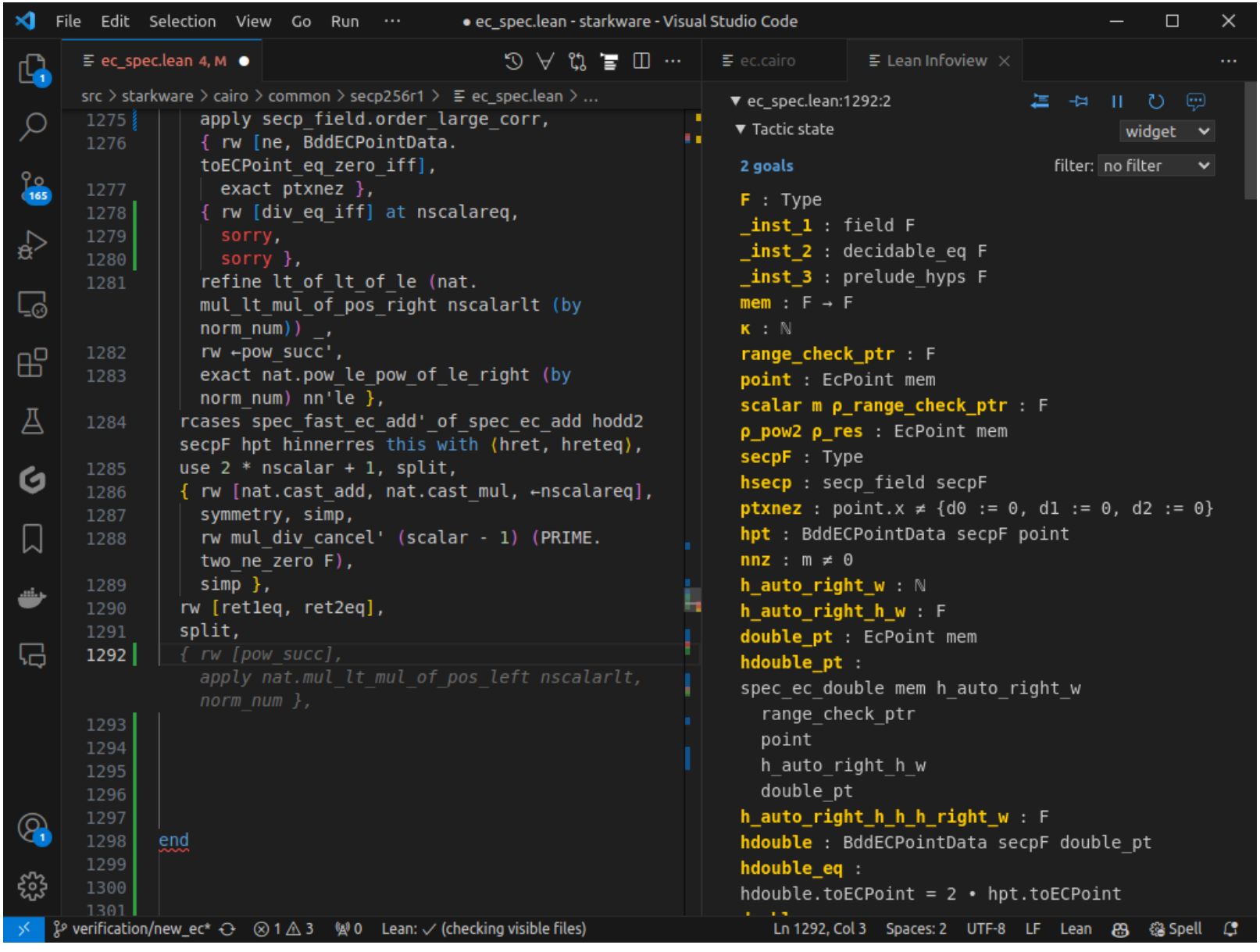
Machine learning tools for Lean

I enjoy Github Copilot.

Machine learning tools for Lean



Machine learning tools for Lean



Outline

- Machine learning and symbolic AI
- Machine learning for mathematics
- Formal methods in mathematics
- Automated reasoning for mathematics
- Automated reasoning in Lean
- Keeping users in mind
- Machine learning and automated reasoning
- Machine learning and formal methods
- Machine learning tools for Lean

Conclusions

To sum up:

- This is an exciting time for mathematics.
- Machine learning and formal methods are still limited, but they are promising.
- We need to find ways to combine the two approaches.
- Benchmarks are important, but machine learning *for* mathematics requires testing in the wild.
- Synthesizing machine learning and symbolic methods is really important.
- Focusing on mathematics is a good way to learn how to do it.