The promise of formal mathematics

Jeremy Avigad

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

Carnegie Mellon University

January 7, 2023

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.

They rely on:

- formal languages
- formal semantics
- formal rules of inference.

Formal methods in mathematics

There are:

- tools for automated reasoning
- tools that support robust user interaction.

Most domains require a combination of the two.

Formal methods can also be used for mathematics.

I will try to explain how, and why they are useful.

Outline

- Formal methods in mathematics
- Interactive theorem provers
- Lean and mathlib
- Why formal methods are useful
- Why logicians should care
- What logicians can contribute

Interactive theorem provers

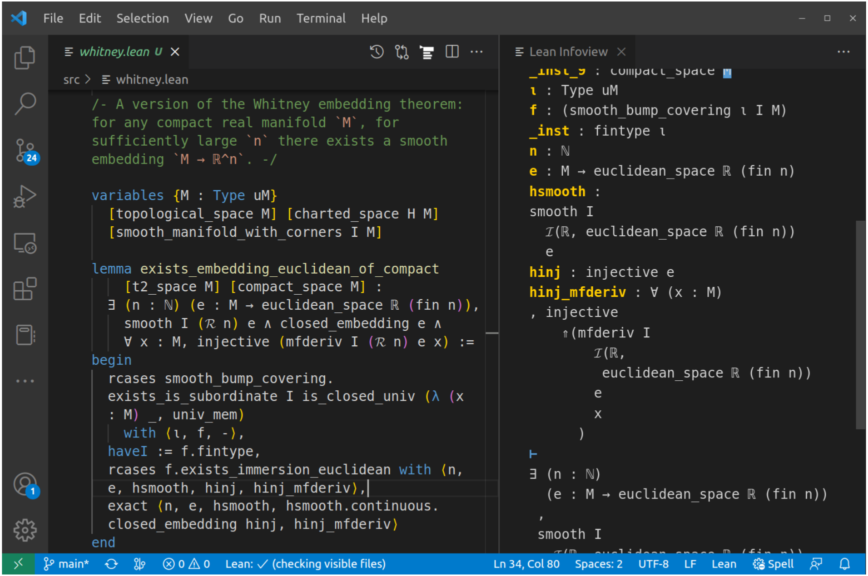
We have known since the early twentieth century that mathematics can be formalized:

- Mathematical statements can be expressed in formal languages, with precise grammar.
- Theorems can be proved from formal axioms, using prescribed rules of inference.

With the help of computational proof assistants, this can be carried out in practice.

In many systems, the formal proof can be extracted and verified independently.

Interactive theorem provers



Interactive theorem provers

"It is not in heaven, that thou shouldest say: 'Who shall go up for us to heaven, and bring it unto us, and make us to hear it, that we may do it?' " (*Deuteronomy* 30:12)

You can download these systems and get started right away.

- Isabelle: `https://isabelle.in.tum.de/`
- Coq with Mathematical Components: `https://math-comp.github.io/`
- HOL Light: `https://www.cl.cam.ac.uk/~jrh13/hol-light/`
- Metamath: `http://us.metamath.org/`
- Lean: `https://leanprover-community.github.io`

There are online documentation, tutorials, user mailing lists, online chat groups, and more.

Interactive theorem provers

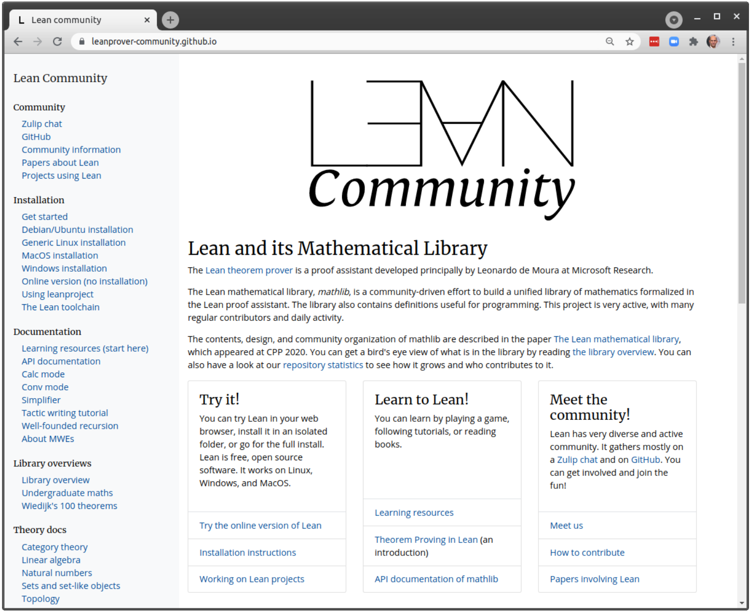
There are a number of systems with substantial mathematical libraries, including Mizar, HOL, Isabelle, Coq, ACL2, PVS, Agda, HOL Light, Metamath, and Lean.

I will focus on Lean because:

- It has received a lot of attention from mathematicians lately.
- It is a system I know particularly well.

This is a snapshot, not a survey.

Lean and mathlib



Lean community

leanprover-community.github.io

Lean Community

Community

- Zulip chat
- GitHub
- Community information
- Papers about Lean
- Projects using Lean

Installation

- Get started
- Debian/Ubuntu installation
- Generic Linux installation
- MacOS installation
- Windows installation
- Online version (no installation)
- Using leanproject
- The Lean toolchain

Documentation

- Learning resources (start here)
- API documentation
- Calc mode
- Conv mode
- Simplifier
- Tactic writing tutorial
- Well-founded recursion
- About MWEs

Library overviews

- Library overview
- Undergraduate maths
- Wiedijk's 100 theorems

Theory docs

- Category theory
- Linear algebra
- Natural numbers
- Sets and set-like objects
- Topology

LEAN Community

Lean and its Mathematical Library

The Lean theorem prover is a proof assistant developed principally by Leonardo de Moura at Microsoft Research.

The Lean mathematical library, *mathlib*, is a community-driven effort to build a unified library of mathematics formalized in the Lean proof assistant. The library also contains definitions useful for programming. This project is very active, with many regular contributors and daily activity.

The contents, design, and community organization of mathlib are described in the paper The Lean mathematical library, which appeared at CPP 2020. You can get a bird's eye view of what is in the library by reading the library overview. You can also have a look at our repository statistics to see how it grows and who contributes to it.

Try it!

You can try Lean in your web browser, install it in an isolated folder, or go for the full install. Lean is free, open source software. It works on Linux, Windows, and MacOS.

- Try the online version of Lean
- Installation instructions
- Working on Lean projects

Learn to Lean!

You can learn by playing a game, following tutorials, or reading books.

- Learning resources
- Theorem Proving in Lean (an introduction)
- API documentation of mathlib

Meet the community!

Lean has very diverse and active community. It gathers mostly on a Zulip chat and on GitHub. You can get involved and join the fun!

- Meet us
- How to contribute
- Papers involving Lean

Lean and mathlib

Lean has been getting good press:

- *Quanta:* "Building the mathematical library of the future"
- *Quanta:* "At the Math Olympiad, computers prepare to go for the gold"
- *Nature:* "Mathematicians welcome computer-assisted proof in 'grand unification' theory"
- *Quanta:* "Proof Assistant Makes Jump to Big-League Math"

Kevin Buzzard gave a talk titled "The Rise of Formalism in Mathematics" at the 2022 International Congress of Mathematicians.

Lean and mathlib

Some achievements:

- a formalization of Ellenberg-Gijswijt cap set theorem (Dahmen, Hölzl, Lewis)
- a formalization of the independence of the continuum hypothesis (Han and van Doorn)
- a formalization of perfectoid spaces (Buzzard, Commelin, and Massot)
- the liquid tensor experiment (Commelin, Topaz, and many others)
- a formalization of Bloom's theorem on unit fractions (Bloom, Mehta)
- a formalization of the sphere eversion theorem (Massot, Nash, and van Doorn)

Lean and mathlib

On December 5, 2020, Peter Scholze challenged anyone to formally verify some of his recent work with Dustin Clausen.

Johan Commelin led the response from the Lean community. On June 5, 2021, Scholze acknowledged the achievement.

"Exactly half a year ago I wrote the Liquid Tensor Experiment blog post, challenging the formalization of a difficult foundational theorem from my Analytic Geometry lecture notes on joint work with Dustin Clausen. While this challenge has not been completed yet, I am excited to announce that the Experiment has verified the entire part of the argument that I was unsure about. I find it absolutely insane that interactive proof assistants are now at the level that within a very reasonable time span they can formally verify difficult original research."

Lean and mathlib

There have been a number of Lean-related meetings, including:

- Lean Together (2019, 2020, 2021)
- Lean for the Curious Mathematician (2020, 2021)
- Learning Mathematics with Lean (2022)
- LeaN in LyoN (2022)

Coming up:

- Machine Assisted Proofs (IPAM)
- Formalization of Cohomology Theories (BIRS)
- Formalization of Mathematics (MSRI summer school)
- Formalization of Mathematics (Copenhagen)
- Machine-Checked Mathematics (Lorentz Center)
- Lean for the Curious Mathematician (CIRM, 2024)

Outline

- Formal methods in mathematics
- Interactive theorem provers
- Lean and mathlib
- Why formal methods are useful
- Why logicians should care
- What logicians can contribute

Why formal methods: verifying correctness

In early 2022, Thomas Bloom solved a problem posed by Paul Erdős and Ronald Graham.

The headline in Quanta read "Math's 'Oldest Problem Ever' Gets a New Answer."

Within in a few months, Bloom and Bhavik Mehta verified the correctness of the proof in Lean.

Why formal methods: verifying correctness

Timothy Gowers
@wtgowers · Jun 13

Very excited that Thomas Bloom and Bhavik Mehta have done this. I think it's the first time that a serious contemporary result in "mainstream" mathematics doesn't have to be checked by a referee, because it has been checked formally. Maybe the sign of things to come ... 1/

> **Kevin Buzzard** @XenaProject · Jun 12
>
> Happy to report that Bloom went on to learn Lean this year and, together with Bhavik Mehta, has now formalised his proof in Lean b-mehta.github.io/unit-fractions/ (including formalising the Hardy-Littlewood circle method), finishing before he got a referee's report for the paper ;-)
>
> Show this thread

2 26 141

Why formal methods: exploring mathematics

Similarly, at the halfway point in the Liquid Tensor experiment, Peter Scholze wrote:

"I am excited to announce that the Experiment has verified the entire part of the argument that I was unsure about."

He went on:

"[H]alf a year ago, I did not understand why the argument worked. . . . "

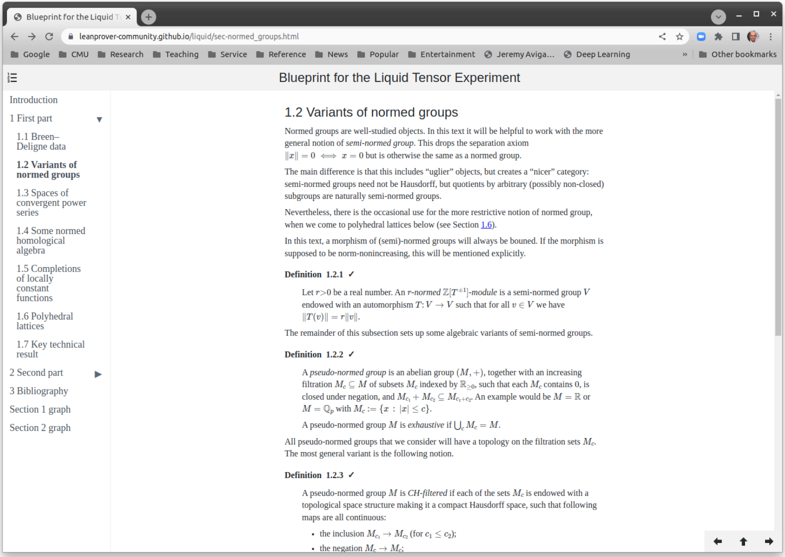
"But during the formalization, a significant amount of convex geometry had to be formalized . . . and this made me realize that . . . the key thing happening is a reduction from a non-convex problem over the reals to a convex problem over the integers."

Why formal methods: collaboration

The liquid tensor experiment is also a model for digital collaboration.

- The formalization was in kept in a shared online repository.
- Participants followed an informal blueprint with links to the repository.
- Participants were in constant contact on Zulip.
- Lean made sure the pieces fit together.

Why formal methods: collaboration



Blueprint for the Liquid T

leanprover-community.github.io/liquid/sec-normed_groups.html

Google CMU Research Teaching Service Reference News Popular Entertainment Jeremy Aviga... Deep Learning » Other bookmarks

Blueprint for the Liquid Tensor Experiment

1.2 Variants of normed groups

Normed groups are well-studied objects. In this text it will be helpful to work with the more general notion of *semi-normed group*. This drops the separation axiom $\|x\| = 0 \iff x = 0$ but is otherwise the same as a normed group.

The main difference is that this includes "uglier" objects, but creates a "nicer" category: semi-normed groups need not be Hausdorff, but quotients by arbitrary (possibly non-closed) subgroups are naturally semi-normed groups.

Nevertheless, there is the occasional use for the more restrictive notion of normed group, when we come to polyhedral lattices below (see Section 1.6).

In this text, a morphism of (semi)-normed groups will always be bouned. If the morphism is supposed to be norm-nonincreasing, this will be mentioned explicitly.

Definition 1.2.1 ✓

Let $r>0$ be a real number. An *r-normed $\mathbb{Z}[T^{\pm 1}]$-module* is a semi-normed group V endowed with an automorphism $T\colon V \to V$ such that for all $v \in V$ we have $\|T(v)\| = r\|v\|$.

The remainder of this subsection sets up some algebraic variants of semi-normed groups.

Definition 1.2.2 ✓

A *pseudo-normed group* is an abelian group $(M, +)$, together with an increasing filtration $M_c \subseteq M$ of subsets M_c indexed by $\mathbb{R}_{\geq 0}$, such that each M_c contains 0, is closed under negation, and $M_{c_1} + M_{c_2} \subseteq M_{c_1+c_2}$. An example would be $M = \mathbb{R}$ or $M = \mathbb{Q}_p$ with $M_c := \{x : |x| \leq c\}$.

A pseudo-normed group M is *exhaustive* if $\bigcup_c M_c = M$.

All pseudo-normed groups that we consider will have a topology on the filtration sets M_c. The most general variant is the following notion.

Definition 1.2.3 ✓

A pseudo-normed group M is *CH-filtered* if each of the sets M_c is endowed with a topological space structure making it a compact Hausdorff space, such that following maps are all continuous:

- the inclusion $M_{c_1} \to M_{c_2}$ (for $c_1 \leq c_2$);
- the negation $M_c \to M_c$;

Why formal methods: teaching

An interactive proof assistant is a powerful tool for teaching mathematics.

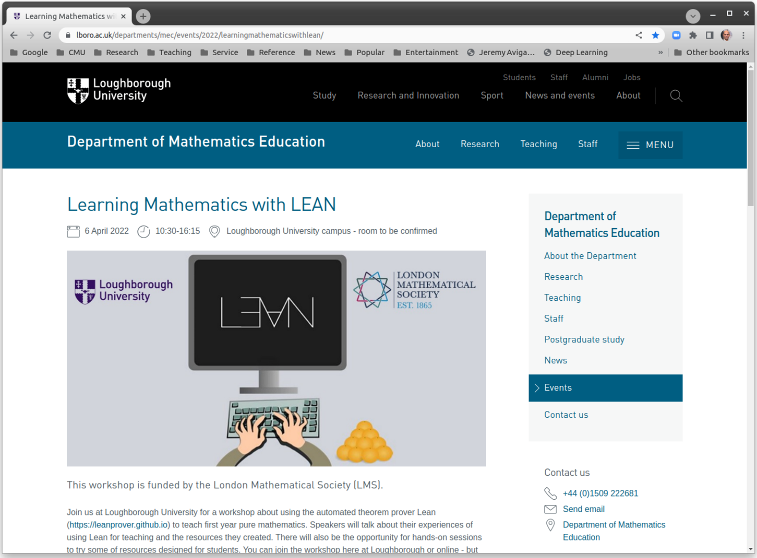
It empowers students to explore mathematical reasoning on their own.

We are starting to see the rise of online communities of people helping each other learn.

We are just beginning to learn how to use the technology effectively.

There have been workshops and conference sessions dedicated to formal methods for teaching.

Why formal methods: teaching



Why formal methods: mathematical computation

A proof assistant can also be used as a platform for numerical and symbolic computation.

A mathematical library in the background provides a precise semantics and a touchstone for interpreting the results.

Tomáš Skřivan has been working on a Lean 4 library for scientific computation.

Alexander Bentkamp, Ramon Fernández Mir, and I have been working on using Lean 4 as a platform for verifying reductions for optimization problems.

Why formal methods: automated reasoning

Automated reasoning tools hold promise for solving combinatorial problems in mathematics.

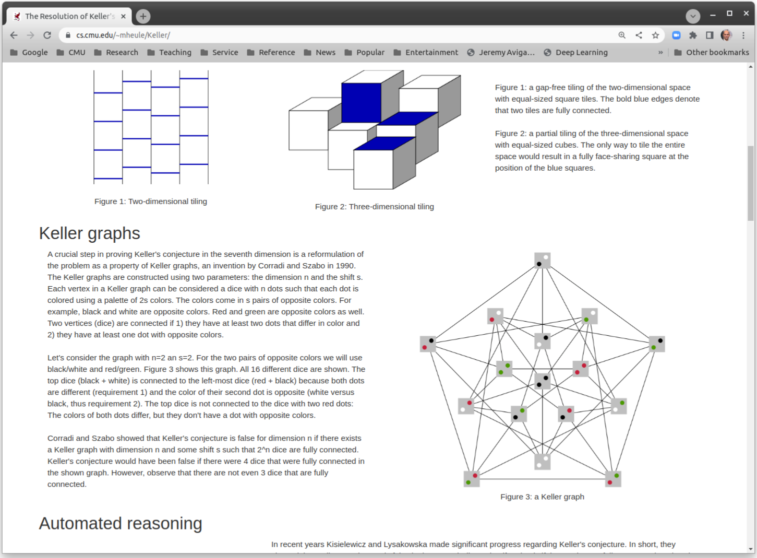
For example, Joshua Brakensiek, Marijn Heule, John Mackey, and David Narváez used a SAT solver to resolve Keller's conjecture:

Quanta, "Computer Search Settles 90-Year-Old Math Problem"

The SAT solver output a proof that was checked with a verified proof checker.

Josh Clune verified the key mathematical reduction in Lean.

Why formal methods: automated reasoning



Why formal methods: machine learning

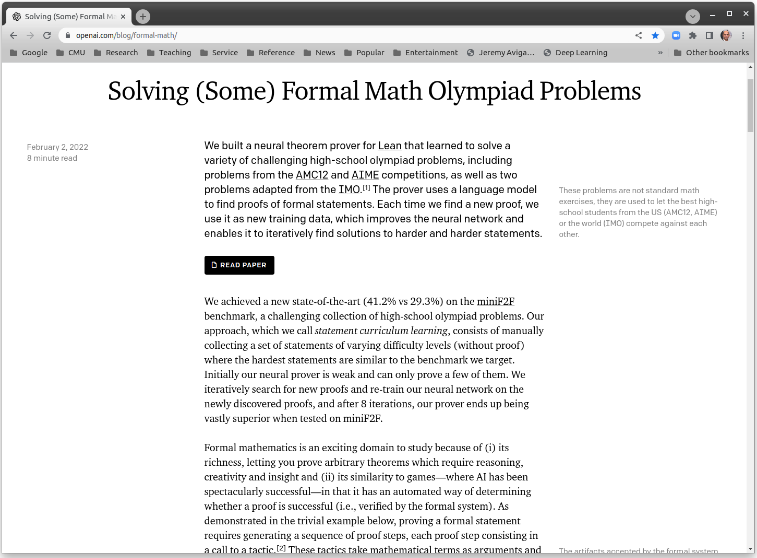
Applications of machine learning to mathematics are a new frontier.

There have been important machine-learning projects using Mizar, HOL Light, Metamath, Isabelle, Coq, Lean, and others.

OpenAI got a neural theorem prover for Lean to solve problems from the International Mathematics Olympiad.

Searching for formally checkable contact provides a clear signal.

Why formal methods: machine learning



Solving (Some) Formal Math Olympiad Problems

February 2, 2022
8 minute read

We built a neural theorem prover for Lean that learned to solve a variety of challenging high-school olympiad problems, including problems from the AMC12 and AIME competitions, as well as two problems adapted from the IMO.[1] The prover uses a language model to find proofs of formal statements. Each time we find a new proof, we use it as new training data, which improves the neural network and enables it to iteratively find solutions to harder and harder statements.

These problems are not standard math exercises, they are used to let the best high-school students from the US (AMC12, AIME) or the world (IMO) compete against each other.

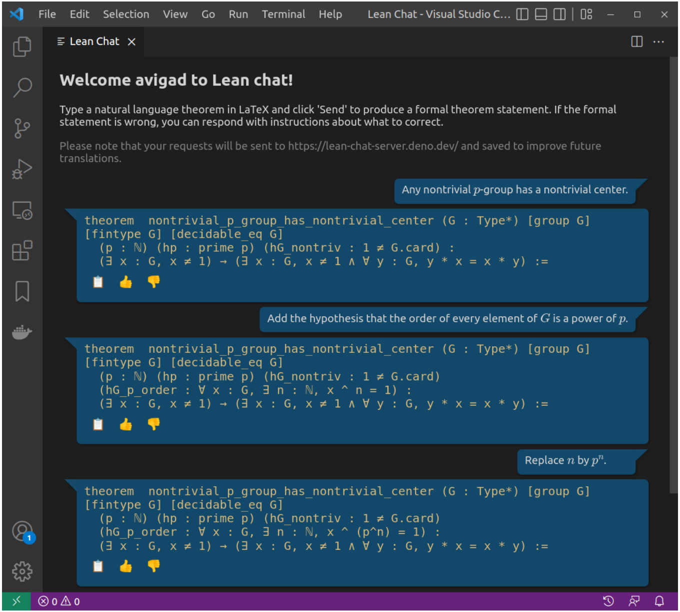
READ PAPER

We achieved a new state-of-the-art (41.2% vs 29.3%) on the miniF2F benchmark, a challenging collection of high-school olympiad problems. Our approach, which we call *statement curriculum learning*, consists of manually collecting a set of statements of varying difficulty levels (without proof) where the hardest statements are similar to the benchmark we target. Initially our neural prover is weak and can only prove a few of them. We iteratively search for new proofs and re-train our neural network on the newly discovered proofs, and after 8 iterations, our prover ends up being vastly superior when tested on miniF2F.

Formal mathematics is an exciting domain to study because of (i) its richness, letting you prove arbitrary theorems which require reasoning, creativity and insight and (ii) its similarity to games—where AI has been spectacularly successful—in that it has an automated way of determining whether a proof is successful (i.e., verified by the formal system). As demonstrated in the trivial example below, proving a formal statement requires generating a sequence of proof steps, each proof step consisting in a call to a tactic.[2] These tactics take mathematical terms as arguments and

The artifacts accepted by the formal system

Why formal methods: machine learning



Why formal methods

Formal technology can help us:

- verify results,
- build mathematical libraries,
- explore new concepts,
- collaborate,
- teach mathematics,
- carry out mathematical computation more rigorously, and
- discover new mathematics.

Outline

- Formal methods in mathematics
- Interactive theorem provers
- Lean and mathlib
- Why formal methods are useful
- Why logicians should care
- What logicians can contribute

Why logicians should care

Formal methods are built on mathematical logic:

- *Deductive systems:* natural deduction, sequent calculi, axiomatic systems
- *Foundations:* set theory, simple type theory, dependent type theory
- *Representations:* formalization, coding, truth, reflection
- *Models of computation:* primitive recursion, type theory, recursion, the lambda calculus
- *Decision procedures:* linear real arithmetic, Presburger arithmetic, real closed fields
- *Proof search:* normal forms, resolution, completeness, Skolemization

Why logicians should care

Mathematics and computer science need each other. Mathematics needs the relevance, and computer science needs the soul.

Formal mathematics is one of the few places where the two communities come together.

The ASL should be there.

What logicians can contribute

From the 1920s to the 1940s, logic developed conceptual foundations for thinking about language and reasoning:

- Formal languages, expressions, and semantics.
- Formal models of computation.

I will discuss five respects in which formal methods today can benefit from a better theoretical understanding.

Mathematical language

Formal logic was designed to model mathematical language.

$$\begin{aligned}&\forall f : \mathbb{R} \to \mathbb{R}\ \forall a, b : \mathbb{R}\\ &\quad (\mathit{continuous}(f) \land a \leq b \land f(a) \leq 0 \land f(b) \geq 0 \to\\ &\qquad \exists x\ (a \leq x \land x \leq b \land f(x) = 0)).\end{aligned}$$

Here is what it looks like in Lean:

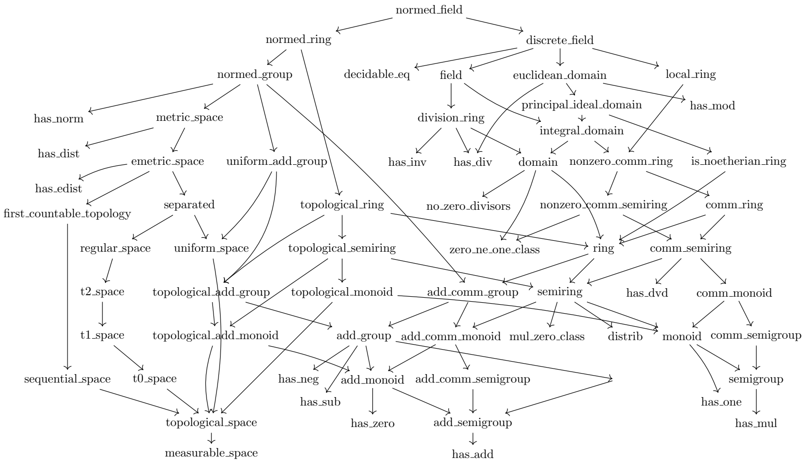
```
∀ f : ℝ → ℝ, ∀ a b : ℝ,
  continuous f → a ≤ b → f a ≤ 0 → f b ≥ 0 →
   ∃ x, a ≤ x ∧ x ≤ b ∧ f x = 0
```

Mathematical language

```
∀ (f : real → real) (a b : real),
@continuous.{0 0} real real
  (@uniform_space.to_topological_space.{0} real
    (@pseudo_metric_space.to_uniform_space.{0} real
   real.pseudo_metric_space))
  (@uniform_space.to_topological_space.{0} real
    (@pseudo_metric_space.to_uniform_space.{0} real
   real.pseudo_metric_space))
  f →
@has_le.le.{0} real real.has_le (f a) (@has_zero.zero.{0} real
   real.has_zero) →
@ge.{0} real real.has_le (f b) (@has_zero.zero.{0} real
   real.has_zero) →
@Exists.{1} real
  (λ (x : real),
    and (@has_le.le.{0} real real.has_le a x)
      (and (@has_le.le.{0} real real.has_le x b) (@eq.{1} real
   (f x) (@has_zero.zero.{0} real real.has_zero))))
```

Mathematical language

In Lean's library *mathlib*, the algebraic hierarchy has hundreds of classes and thousands of instances.



Mathematical language

Type classes are used for notation, bookkeeping (decidable types, inhabited types, coercions), order structures, linear algebra, topological spaces, category theory, function spaces (inner product spaces, normed spaces), measure theory, manifolds, computability, and more.

There are tons of dependencies between them.

The real numbers are simultaneously an instance of a field, an ordered field, a normed field, a metric space, a topological space, a uniform space, a vector space (over the reals), a manifold, a measure space, ...

Mathematical language

Conceptual question: is there room for a theory of mathematical language that tells us how mathematical language really works?

Challenges:

- Understanding how we leave information implicit.
- Understanding how we overload notation.
- Understanding how we resolve ambiguities.
- Understanding how we establish canonical interpretations.
- Understanding how we avoid conflicts.
- Understanding how we identify objects that are really different.
- Understanding how we do all this so quickly.

Mathematical representations

Consider two different ways to represent a morphism that preserves multiplication.

```
structure mul_hom (M : Type*) (N : Type*)
  [has_mul M] [has_mul N] :=
(to_fun : M → N)
(map_mul : ∀ x y, to_fun (x * y) = to_fun x * to_fun y)

structure is_mul_hom {α β : Type*} [has_mul α] [has_mul β]
  (f : α → β) : Prop :=
(map_mul : ∀ x y, f (x * y) = f x * f y)
```

Mathlib initially favored unbundled morphisms, but then, in 2019, switched to bundled morphisms.

Anne Baanen has proposed a method of getting the best of both worlds.

Mathematical representations

Another example: consider field extensions $E \subseteq F \subseteq K$.

Working formally, it is often better to use independent data types rather than subsets.

A better idea: reason about embeddings $E \hookrightarrow F \hookrightarrow K$.

An even better idea: reason about F as an E-algebra, K as an F-algebra, and K as an E-algebra, with a coherence condition on scalar multiplication.

The class field theory library is built on these insights.

Mathematical representations

There is a sense in which all this is trivial. Mathematicians *know* that a structural viewpoint is important.

But there is a value to making implicit knowledge explicit and engineering representations so that they fit together nicely and support a much larger edifice.

Conceptual question: is there a mathematical theory that can help us understand how we choose representations and organize knowledge so that:

- communication is efficient
- reasoning is efficient
- reasoning is reliable.

Mathematical inference

Automated reasoning is a vast industry.

There are decision procedures, constraint solvers, SAT solvers, SMT solvers, model checkers, equational theorem provers, term rewriters, first-order theorem provers, model finders, higher-order theorem provers, relevance filters, sledgehammers, and more.

Automated procedures are good at large, homogeneous inferences, but not so good at using ordinary mathematical expertise.

Filling in straightforward textbook inferences is often inordinately painful.

Mathematical inference

Jiannis Limperg and Asta Halkyær have developed automation for Lean called AESOP, which stands for "Automated Extensible Search for Obvious Proofs."

We need a theory of the obvious.

Conceptual question: is there a theory of mathematical reasoning that can explain what makes a straightforward inference straightforward?

It needs to account for mathematical expertise, domain-general and domain-specific cues and heuristics to find the relevant facts and inferences.

Reliable knowledge

Formal proof is an ideal. Real mathematical knowledge is messy.

What is the relationship between ordinary mathematical practice and the formal ideal?

Conceptual question: why is mathematics formalizable? How does our informal mathematics manage to track the formal ideal?

(See my paper, "The reliability of mathematical inference.")

Reliable knowledge

People working in formal methods are very sensitive to what is being verified and what is being trusted (the "trust story"). It's a form of recreational paranoia raised to a high art.

We place trust in axiomatic foundations, specifications, implementations, and hardware. There are ways to minimize likelihood of error.

What do we trust when we use formal methods to verify complex systems like self driving cars, airline control systems, operating systems, and so on?

What ensures the reliability of mathematical arguments, and what ensures the reliability of the application of mathematical results?

Symbolic methods

There is a tension between symbolic methods ("good old fashioned AI") and machine learning.

With all the impressive successes of neural networks, do symbolic methods still have a role to play?

There is interest in *explainable AI*: getting ML systems to explain and justify their conclusions.

Putting it that way makes the explanations sound like an afterthought.

Symbolic methods

Searching for mathematical proofs involves searching for something formal and precise.

Conceptual questions: Is there an *intrinsic* value to symbolic expressions and representations? Are there problems we want to solve for which symbolic methods are ineliminable?

Mathematics has a strong aesthetic value, but can we say more?

In light of modern AI, what role should mathematical reasoning play in the way we conceptualize the world?

What logicians can contribute

In short, we need to understand:

- the nature of mathematical language
- the nature of mathematical representations
- the nature of mathematical inference
- the nature of mathematical knowledge
- the proper and reliable warrants for mathematical knowledge (and other types of knowledge that depend on it)
- the relationship between mathematical knowledge and other types of knowledge.

Conclusions

Formal methods have a lot to offer mathematics.

The field is young, and we have a lot to learn.

We need theory as well as experimentation.

Mathematical logic can play a role.

Challenge question: who wrote this?

"It has long been recognized that mathematics and logic are virtually the same and that they may be expected to merge imperceptibly into one another. Actually this merging process has not gone at all far, and mathematics has profited very little from researches in symbolic logic. The chief reasons for this seem to be a lack of liaison between the logician and the mathematician-in-the-street. Symbolic logic is a very alarming mouthful for most mathematicians, and the logicians are not very much interested in making it more palatable. It seems however that symbolic logic has a number of small lessons for the mathematician which may be taught without it being necessary for him to learn very much of symbolic logic."