

Mathematics in Lean

Floris van Doorn

University of Paris-Saclay
soon: University of Bonn

September 18, 2023

Computers and mathematics

Can a computer help a mathematician with mathematical reasoning?

Computers and mathematics

Can a computer help a mathematician with mathematical reasoning?

Computers are used in mathematics to compute.

- Run simulations
- Test conjectures
- Compute number sequences, knot invariants ...

Computers and mathematics

Can a computer help a mathematician with mathematical **reasoning**?

Computers are used in mathematics to compute.

- Run simulations
- Test conjectures
- Compute number sequences, knot invariants ...

An **proof assistant** or **interactive theorem prover** is a program that can reason with mathematical definitions, theorems and proofs, provided a user writes these in a language that the program can understand.

This is called **formalization**.

Overview of the talk

- Lean demo
- Some history of formalized math
- Formalization in Lean
- Why formalization?

Lean is an interactive theorem prover developed by Leonardo de Moura at Microsoft Research.



It is based on type theory.

It is open source, and under active development for over 10 years.

Let's see it in action.

Overview of the talk

- Lean demo
- Some history of formalized math
- Formalization in Lean
- Why formalization?

Formalized math before 2000

- 1960s: Automath
- 1970s: Mizar
- 1980s: HOL family, Isabelle, Coq
- 1990s: proofs of basic undergraduate theorems could be checked by the computer


1960s: Automath

```
A * NMP      :=      ---      # NOT(<M>P)
NMP * N      :=      ---      # NAT
N * NP       :=      ---      # <N>P
NP * T16     := MP(<N>P, LESSIS(M,N), NP, <N>T14) # LESSIS(M,N)
NP * T17     := TH3*L-IMP*(IS(M,N), <M>P, NMP,
      EX, IS(M,N))ISP(NAT,P,N,M,NP,
      SYMIS(NAT,M,N,X)) # NOT(IS(M,N))
NP * T18     := ORE1(LESS(M,N), IS(M,N), T16,
      T17) # LESS(M,N)
NP * T19     := SATZ25B(N,M,T18) # LESSIS(PL(M,1),N)
NMP * T20    := [X,NAT]Y,<X>P]T19(X,Y) # LB(PL(M,1))
NMP * T21    := MP(LB(PL(M,1)), CON, T20, T15) # CON
A * T22     := ET(<M>P, [X,NOT(<M>P)]T21(X)) # <M>P
A * T23     := ANDI(LB(M), <M>P, T14, T22) # MIN(M)
```

-327

```
S * SATZ27   := TH6*L-SOME*(NAT,[X,NAT]
      AND(LB(X), NOT(LB(PL(X,1))))),
      EX,NAT]MIN(X), T13*-327*,
      [X,NAT]Y, AND(LB(X),
      NOT(LB(PL(X,1))))]T23*-327*(X,
      Y) # SOME(EX,NAT]MIN(P,X))
```

1970s: Mizar

::  WP: The Infinitude of Primes

theorem *Th79*: :: *NEWTON:79*

SetPrimes is infinite

proof

assume *A1*: SetPrimes is finite ; :: *thesis*:

then reconsider *SP* = SetPrimes as finite set ;

consider *n* being Nat such that

A2: SetPrimes , Seg *n* are_equipotent by *A1*, *FINSEQ_1:56*;

card SetPrimes = card (Seg *n*) by *A2*, *CARD_1:5*;

then card SetPrimes = card *n* by *FINSEQ_1:55*;

then *A3*: card *SP* = *n* ;

set *p* = primenumber (*n* + 1);

set *Spp* = SetPrimenumber (primenumber (*n* + 1));

A4: *n* + 1 = card (SetPrimenumber (primenumber (*n* + 1))) by *Def8*;

card (SetPrimenumber (primenumber (*n* + 1))) <= card *SP* by *Th68*, *NAT_1:43*;

hence contradiction by *A3*, *A4*, *NAT_1:13*; :: *thesis*:

end;

Formalized math after 2000

- 2005: Georges Gonthier et al. formalizes the four color theorem in Coq
- 2005: Jeremy Avigad et al. formalizes the prime number theorem in Isabelle
- 2012: Georges Gonthier et al. formalizes the Feit-Thompson theorem in Coq
- 2014: Tom Hales et al. formalizes his proof of the Kepler conjecture

Overview of the talk

- Lean demo
- Some history of formalized math
- **Formalization in Lean**
- Why formalization?

mathlib

mathlib is the mathematical library of Lean.

It is a **general-purpose**: it contains algebra, topology, analysis, differential geometry, probability theory, category theory, combinatorics, logic, ...

It is **decentralized**: contributors come with their own plans and goals.

It is **large**: mathlib has over 1 million lines of code, written by over 270 contributors.

It is **active**: There are more than 100 contributions every week.

It is **coherent**: Every contribution is reviewed by one of the 38 reviewers, and the quality of the library is ensured by the 26 mathlib maintainers.

I have worked with Lean since 2014 and am a maintainer since 2019.

Selected Lean formalizations

- Spectral sequences (2017; van Doorn et al.)
- Definition of perfectoid spaces (2019; Buzzard, Commelin, Massot)
- Independence of the continuum hypothesis (2019; van Doorn, Han)
- Cap set problem (2019; Dahmen, Hölzl, Lewis)
- Witt vectors (2020; Commelin, Lewis)
- Finiteness of the class group of a global field (2021; Baanen, Dahmen, Narayanan, Nuccio)
- Liquid tensor experiment (2022; Commelin et al.)
- Erdős-Graham problem (2022; Bloom and Metha)
- Sphere eversion project (2022; van Doorn, Massot, Nash)

Overview of the talk

- Lean demo
- Some history of formalized math
- Formalization in Lean
- **Why formalization?**

Checking proofs

We won't be able to formalize all research mathematics any time soon.

However, we can formalize the parts of proofs that are hard to check by humans:

- proofs that are very long
- proofs that are very technical

Teaching Mathematics

Proof assistant are used in some universities to teach undergraduate mathematics.

Requiring students to write proofs in a proof assistant forces them to write structures proofs.

The students get immediate feedback from the proof assistant.

Explore mathematics

Formalization gives new ways to explore mathematics.

Kyle Miller and Patrick Massot are working on an **informalizer**. This turns a formal proof into a readable proof, where the reader decides how much detail they want.

Explore mathematics

Formalization gives new ways to explore mathematics.

Kyle Miller and Patrick Massot are working on an **informalizer**. This turns a formal proof into a readable proof, where the reader decides how much detail they want.

Some people in the AI community think that AI will be better than humans at mathematical research within 10 years.

I am very skeptical of this claim, but I think AI might help us explore mathematics.

Example: generalizing a definition

Formalization can give interesting generalizations.

Example: generalizing a definition

Formalization can give interesting generalizations.

Given $\varphi : \mathbb{R}^n \rightarrow \mathbb{R}$ and $f : \mathbb{R}^n \rightarrow \mathbb{R}^m$ we define the **convolution**

$$(\varphi * f)(x) = \int \varphi(t) f(x - t) dt.$$

We have (under suitable conditions)

$$\frac{\partial}{\partial x_i} (\varphi * f) = \frac{\partial \varphi}{\partial x_i} * f.$$

It would be nice to have a formula for the total derivative.

$D(\varphi * f) = D\varphi * f$ doesn't work since neither $D\varphi$ nor f is scalar-valued.

Example: generalizing a definition

Formalization can give interesting generalizations.

Given $\varphi : \mathbb{R}^n \rightarrow \mathbb{R}$ and $f : \mathbb{R}^n \rightarrow \mathbb{R}^m$ we define the **convolution**

$$(\varphi * f)(x) = \int \varphi(t) f(x - t) dt.$$

We have (under suitable conditions)

$$\frac{\partial}{\partial x_i} (\varphi * f) = \frac{\partial \varphi}{\partial x_i} * f.$$

It would be nice to have a formula for the total derivative.

$D(\varphi * f) = D\varphi * f$ doesn't work since neither $D\varphi$ nor f is scalar-valued.

We can use a bilinear map L to define the convolution:

$$(\varphi *_L f)(x) = \int L(\varphi(t), f(x - t)) dt$$

Now we have (under suitable conditions)

$$D(\varphi *_L f) = D\varphi *_L f$$

where L' is defined by $L'(A, y)(v) = L(Av, y)$.

Collaboratively creating a mathematical library

There is lots of activity in creating a formalized mathematical library collaboratively.

There is lots of discussions, and people reviewing each other's work.

It is an exciting project and you can contribute!

Getting involved

You cannot learn Lean only by listening to a talk or reading a textbook.

This week you can learn Lean by **active learning**: writing proofs yourself.

You can continue learning Lean after this week, using the (WIP) text book **Mathematics in Lean** and doing its exercises:

https://leanprover-community.github.io/mathematics_in_lean/

You can ask questions on

<https://leanprover.zulipchat.com/>

You are encouraged to start your own project and formalize something simple in your area of mathematics.

If you want to contribute: make sure it makes its way into `mathlib`.

Why formalize mathematics?

- 1 Collaboratively create a unified mathematical library
- 2 Check proofs
- 3 Teach mathematics
- 4 Explore mathematics
- 5 Create mathematics

ATP and machine learning

Interactive theorem proving means that the user writes the mathematical proof, checked by the proof assistant.

- ▶ Kyle Miller and I will give a tutorial on interactive theorem proving in Lean.
- ▶ Jeremy Avigad will lecture about the logical foundations of proof assistants.

Automated theorem proving means that the user writes the theorem statement, and the computer finds a proof by itself.

- ▶ Alexander Bentkamp will talk about this.

Recently groups at Google Deepmind and Meta AI have used **machine learning** to let computers find proofs themselves.

- ▶ Adam Zslot Wagner will talk about this.

Conclusion

- Lean is a proof assistant with a lot of exciting ongoing projects.
- You can contribute to formalized mathematics.
- We don't really know exactly how formalization will change mathematics, but it will be exciting.