Learning Lean Seminar

Matej Penciak

Northeastern University

February 4, 2022
Slides available at:
https://github.com/mpenciak/Lean-Seminar-Sp2022

What is Lean?

What is Lean?

“Lean is a functional programming language that makes it easy to write correct and maintainable code. **You can also use Lean as an interactive theorem prover**. Lean programming primarily involves defining types and functions.”
(https://leanprover.github.io/about/)

What is an interactive theorem prover/proof assistant?

What is an interactive theorem prover/proof assistant?

It's a tool to that helps bridge the gap between proofs that can be understood by humans, and proofs that can be understood by computers.

What is an interactive theorem prover/proof assistant?

It's a tool to that helps bridge the gap between proofs that can be understood by humans, and proofs that can be understood by computers.

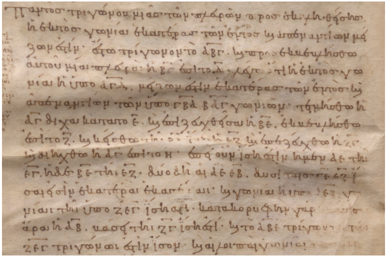
This answer is purposefully vague, and before we can make it more precise we need to first understand what constitutes a mathematical proof.

What is a proof?

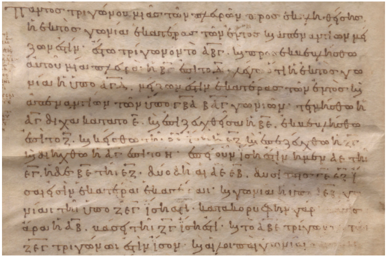
What is a proof?

The answer to this is tricky, and the definition has changed over the course of the history of mathematics!

Proofs in ancient Greece



Proofs in ancient Greece



Prop. 16: Upon one of the sides of any triangle being extended, the external angle is larger than each of the interior and opposite angles.

Let there be a triangle, ABG, and let one side of it, BG, be extended to D. I say that the exterior angle, that by AGD, is larger than each of the interior and opposite angles, the angles by GBA, BAG. Let AG be bisected at E, and let BE, being joined, be extended on a straight-line to Z, and let . . .

Proofs in the age of enlightenment

LEMMA XVI.

From three given points to draw to a fourth point that is not given three right lines whoſe differences ſhall be either given or none at all.

CASE I. Let the given points be *A*, *B*, *C*, (*Pl.* 8. *Fig.* 1.) and *Z* the fourth point which we are to find; becauſe of the given difference of the lines *AZ*, *BZ*, the locus of the point *Z* will be an hyperbola, whoſe foci are *A* and *B*, and whoſe principal axe is the given difference. Let that axe be *MN*. Taking *PM* to *MA*, as *MN* is to *AB*, erect *PR* perpendicular to *AB*, and let fall *ZR* perpendicular to *PR*; then, from the nature of the hyperbola, *ZR* will be to *AZ* as *MN* is to *AB*. And by the like argument, the locus of the point *Z* will be another hyperbola, whoſe foci are *A*, *C*, and whoſe principal axe is the difference between *AZ* and *CZ*; and *QS* a perpendicular on *AC* may be drawn, to which (*QS*) if from any point *Z* of this hyperbola

Proofs in the early 20th century

dine m spezzate in α e in una superficie di ordine $m-1$.

Se C_0 è generica, siccome per $m \geq 4$ le superficie di Σ birazionalmente equivalenti, e quindi omografiche, alla generica di esse son ∞^4 fra loro omologiche (n. 9), la superficie variabile in Σ dipende da $\delta_0 = \binom{m+2}{3} - 4$ moduli.

Se C_0 si particolarizza, la dimensione μ di Σ rimane immutata e la dimensione della varietà delle superficie di Σ birazionalmente equivalenti alla generica superficie di Σ si conserva non inferiore a 4, perché ogni superficie di Σ è portata in ∞^4 altre dalle omologie che hanno per piano d'omologia α. Pertanto anche quando C_0 è particolare non vi sono nella superficie variabile in Σ più di δ_0 moduli. Ne segue che:

Il numero δ dei nodi di una superficie di ordine m dello spazio ordinario, non avente altre singolarità, soddisfa alla limitazione

$$(1) \qquad \delta \leq \lambda - 4 \qquad \left[\lambda = \binom{m+2}{3}\right].$$

Proofs now

Proposition 4. *The vector fields of the framed RS hierarchy of Definition 6 agree with the Hamiltonian vector fields defined by the map* **H**.

Proof. We will prove the theorem by calculating the flows given by homogeneous elements of $\mathrm{Hitchin}_{RS}(E)$. Fix a $\xi \in H^0(E, \mathcal{O}_E(D_0 + D_\infty)^i)^* \cong H^1(E, \mathcal{O}_E(-D_0 - D_\infty)^i)$ where the duality is given by the usual residue pairing. Therefore the function on $\mathsf{RS}_{\sigma,n}(E, V)$ defined by ξ is given by

$$H_\xi(W, \eta_0, \eta_\infty) = \operatorname{Res}\left(\xi \operatorname{Tr}\left(\frac{1}{i+1}(\eta_0 \cdot \eta_\infty)^{i+1}\right)\right).$$

Proofs now

The following result follows immediately from the discussion of Definition 4.1, (ii).

Proposition 4.2. **(The Set of Label Classes of Cusps of a Base-Prime-Strip)** *Let* $^{\dagger}\mathfrak{D} = \{^{\dagger}\mathcal{D}_{\underline{v}}\}_{\underline{v}\in\underline{\mathbb{V}}}$ *be a* $\mathcal{D}$**-prime-strip**. *Then for any* $\underline{v}, \underline{w} \in \underline{\mathbb{V}}$*, there exist* **bijections**

$$\mathrm{LabCusp}(^{\dagger}\mathcal{D}_{\underline{v}}) \xrightarrow{\sim} \mathrm{LabCusp}(^{\dagger}\mathcal{D}_{\underline{w}})$$

that are **uniquely determined** *by the condition that they be* **compatible** *with the assignments* $^{\dagger}\underline{\eta}_{\underline{v}} \mapsto {}^{\dagger}\underline{\eta}_{\underline{w}}$ *[cf. Definition 4.1, (ii)], as well as with the* $\mathbb{F}_l^{\circledast}$**-torsor structures** *on either side. In particular, these bijections are* **preserved** *by arbitrary* **isomorphisms** *of* $\mathcal{D}$*-prime-strips. Thus, by identifying the various "*$\mathrm{LabCusp}(^{\dagger}\mathcal{D}_{\underline{v}})$*" via these bijections, it* **makes sense to write** $\mathrm{LabCusp}(^{\dagger}\mathfrak{D})$.

Proofs in the future?

```lean
/-- Nonzero integral ideals in a Dedekind domain are invertible.

We will use this to show that nonzero fractional ideals are invertible,
and finally conclude that fractional ideals in a Dedekind domain form a group with zero.
-/
lemma coe_ideal_mul_inv [h : is_dedekind_domain A] (I : ideal A) (hI0 : I ≠ ⊥) :
  (I * I⁻¹ : fractional_ideal A⁰ K) = 1 :=
begin
  -- We'll show `1 ≤ J⁻¹ = (I * I⁻¹)⁻¹ ≤ 1`.
  apply mul_inv_cancel_of_le_one hI0,
  by_cases hJ0 : (I * I⁻¹ : fractional_ideal A⁰ K) = 0,
  { rw [hJ0, inv_zero'], exact fractional_ideal.zero_le _ },
  intros x hx,
  -- In particular, we'll show all `x ∈ J⁻¹` are integral.
  suffices : x ∈ integral_closure A K,
  { rwa [is_integrally_closed.integral_closure_eq_bot, algebra.mem_bot, set.mem_range,
         ← fractional_ideal.mem_one_iff] at this;
      assumption },
  -- For that, we'll find a subalgebra that is f.g. as a module and contains `x`.
  -- `A` is a noetherian ring, so we just need to find a subalgebra between `{x}` and `I⁻¹`.
  rw mem_integral_closure_iff_mem_fg,
  have x_mul_mem : ∀ b ∈ (I⁻¹ : fractional_ideal A⁰ K), x * b ∈ (I⁻¹ : fractional_ideal A⁰ K),
  { intros b hb,
    rw mem_inv_iff at ⊢ hx,
```

What is proof formalization?

In a very simplified form, Lean has a small kernel of the essential rules of logical deduction.

What is proof formalization?

In a very simplified form, Lean has a small kernel of the essential rules of logical deduction.

The goal of proof formalization is to implement the proofs of all of the major mathematical results starting from these basic principles.

What is proof formalization?

In a very simplified form, Lean has a small kernel of the essential rules of logical deduction.

The goal of proof formalization is to implement the proofs of all of the major mathematical results starting from these basic principles.

The ongoing proof formalization project in Lean is named mathlib (https://github.com/leanprover-community/mathlib)

What is proof formalization?

In a very simplified form, Lean has a small kernel of the essential rules of logical deduction.

The goal of proof formalization is to implement the proofs of all of the major mathematical results starting from these basic principles.

The ongoing proof formalization project in Lean is named mathlib (https://github.com/leanprover-community/mathlib)

This should not be confused with proof by checking all the cases! (Like the proof of the 4 color theorem)

Why formalize proofs?

1. Catch any errors we made along the way (not as many as you'd think)

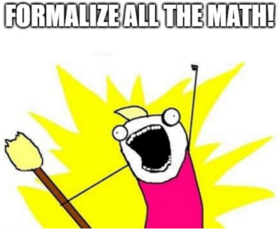
Why formalize proofs?

1. Catch any errors we made along the way (not as many as you'd think)
2. Develop insights into how results are proven. A once in human history chance to re-factor all of mathematics!

Why formalize proofs?

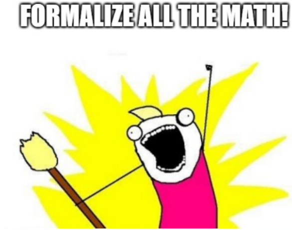
1. Catch any errors we made along the way (not as many as you'd think)
2. Develop insights into how results are proven. A once in human history chance to re-factor all of mathematics!
3. Help future mathematicians to prove new theorems by developing tools along the way.

Formalize all the math!



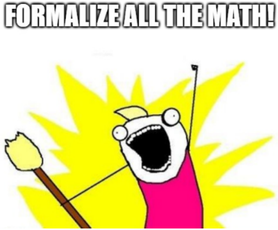
Q: Does that mean we have to implement every theorem into Lean by hand?

Formalize all the math!



Q: Does that mean we have to implement every theorem into Lean by hand?
A: (short answer) Yes!

Formalize all the math!



Q: Does that mean we have to implement every theorem into Lean by hand?
A: (short answer) Yes!
A: (long answer) Probably! At least for now.

Other interactive theorem provers

1. Agda
2. Coq
3. HOL
4. Idris
5. ... Many more!

Other interactive theorem provers

1. Agda
2. Coq
3. HOL
4. Idris
5. ... Many more!

Why Lean then?

Other interactive theorem provers

1. Agda
2. Coq
3. HOL
4. Idris
5. … Many more!

Why Lean then?

Lean just feels right for what I (a mathematician) wants to do (prove theorems).

Other interactive theorem provers

1. Agda
2. Coq
3. HOL
4. Idris
5. ... Many more!

Why Lean then?

Lean just feels right for what I (a mathematician) wants to do (prove theorems). A specific example is quotients!

Other interactive theorem provers

1. Agda
2. Coq
3. HOL
4. Idris
5. … Many more!

Why Lean then?

Lean just feels right for what I (a mathematician) wants to do (prove theorems). A specific example is quotients! And sheaves

Other interactive theorem provers

1. Agda
2. Coq
3. HOL
4. Idris
5. ... Many more!

Why Lean then?

Lean just feels right for what I (a mathematician) wants to do (prove theorems). A specific example is quotients! And sheaves and schemes

Other interactive theorem provers

1. Agda
2. Coq
3. HOL
4. Idris
5. ... Many more!

Why Lean then?

Lean just feels right for what I (a mathematician) wants to do (prove theorems). A specific example is quotients! And sheaves and schemes and triangulated categories

Other interactive theorem provers

1. Agda
2. Coq
3. HOL
4. Idris
5. ... Many more!

Why Lean then?

Lean just feels right for what I (a mathematician) wants to do (prove theorems). A specific example is quotients! And sheaves and schemes and triangulated categories and...

Other interactive theorem provers

1. Agda
2. Coq
3. HOL
4. Idris
5. ... Many more!

Why Lean then?

Lean just feels right for what I (a mathematician) wants to do (prove theorems). A specific example is quotients! And sheaves and schemes and triangulated categories and...

The biggest part is the community! So many resources to learn from, and everyone is so friendly and so excited, it's hard not to get swept up in the hype!

Where do I fit into all this?

There's a lot of math, so everyone little bit helps!

Where do I fit into all this?

There's a lot of math, so everyone little bit helps!

There's a lot of parts of the undergraduate curriculum that have yet to be formalized.

Where do I fit into all this?

There's a lot of math, so everyone little bit helps!

There's a lot of parts of the undergraduate curriculum that have yet to be formalized.

Might as well start now.

Plan for the semester

1. Learn about the Lean platform by attending short talks like this, and working in groups on small projects
2. Learn about specific parts of the mathlib (kind of like learning a new API or module)
3. Pick a theorem, and try to try to formalize and work in groups to add it to mathlib!

Plan for the semester

1. Learn about the Lean platform by attending short talks like this, and working in groups on small projects
2. Learn about specific parts of the mathlib (kind of like learning a new API or module)
3. Pick a theorem, and try to try to formalize and work in groups to add it to mathlib!
4. (for mathematicians) Learn all of the useful tools that computer scientists have been hiding from us (version control with git, SAT solvers)
5. (for computer scientists) Maybe learn some math along the way?

Enough slides, lets prove some theorems!