Formalizing sphere eversion in Lean

Floris van Doorn

University of Paris-Saclay

9 May 2023

Talk Topics

In this talk:

1. Lean and mathlib
2. The sphere eversion project
3. A convenient formalization of vector bundles

Lean

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

It is open source, and under active development for over 9 years.

Let's see it in action.

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 and every contribution is reviewed by one of the 23 mathlib maintainers.

I have worked with Lean since 2014 and am a mathlib 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)
- 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.)
- Sphere eversion project (2022; van Doorn, Massot, Nash)

Lean 3 vs Lean 4

- 2018: The Lean core developers started working on Lean 4
- 2021: First milestone release of Lean 4
- 2022: Lean 4 has all essential features of Lean 3
- Oct 2022 - present: port mathlib to Lean 4
  - Partially automated, with many manual fixes
  - Various problems arose from differences between Lean 3 and Lean 4
  - Currently 60% has been ported (615k out of 1020k lines)

Why Lean 4?

- Turn Lean into a fully-fledged programming language, compiled into C
- Rewrite Lean in Lean
- All trusted functions are total, but unsafe functions can use general recursion, foreign functions and unsafe features like pointer equality.
- Flexible macro expansion

```
syntax "{ " ident (" : " term)? " // " term " }" : term
macro_rules
  | `({ $x : $type // $p }) => ``(Subtype (fun ($x:ident : $type) => $p))
#check { x : ℕ // x < 10 }
```

- Convenient do notation

```
def List.findM? (p : α → m Bool)
  (xs : List α) : m (Option α) := do
  for x in xs do
    let b ← p x
    if b then
      return some x
  return none
```

- And many more features!

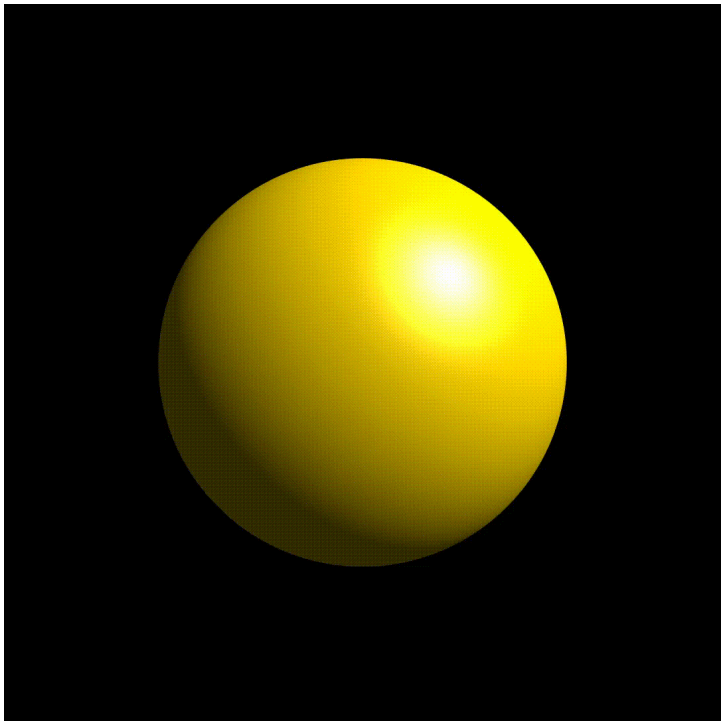
Talk Topics

In this talk:

1. Lean and mathlib
2. The sphere eversion project
3. A convenient formalization of vector bundles

Sphere eversion

We can turn a sphere inside out without tearing or creasing it, but allowing self-intersections.



To precisely state this,
we use the following definition:
f is an immersion $\iff$
f is locally an embedding $\iff$
the total derivative of f is injective.

Sphere eversion

Theorem (Smale, 1957)

There is a smooth transformation of immersions

$$\mathbb{S}^2 \to \mathbb{R}^3$$

starting with the inclusion map and ending with the antipodal map.

We don't give an explicit construction, instead we use a technique called convex integration.

Convex integration (1)

Suppose we want to turn $f : \mathbb{R}^2 \to \mathbb{R}^3$ into an immersion.

Convex integration (1)

Suppose we want to turn $f : \mathbb{R}^2 \to \mathbb{R}^3$ into an immersion.

We can do this by first ensuring that $\partial_1 f(x) := \frac{\partial f(x)}{\partial x_1} \neq 0$ and next that $\partial_2 f(x)$ is not collinear with $\partial_1 f(x)$

In both steps we want that $\partial_j f(x)$ lives in some open subset $\Omega_x \subseteq \mathbb{R}^3$.

Convex integration (1)

Suppose we want to turn $f : \mathbb{R}^2 \to \mathbb{R}^3$ into an immersion.

We can do this by first ensuring that $\partial_1 f(x) := \frac{\partial f(x)}{\partial x_1} \neq 0$ and next that $\partial_2 f(x)$ is not collinear with $\partial_1 f(x)$

In both steps we want that $\partial_j f(x)$ lives in some open subset $\Omega_x \subseteq \mathbb{R}^3$.

Suppose there exists a family of loops $\gamma : \mathbb{R}^2 \times \mathbb{S}^1 \to \mathbb{R}^3$ such that γ_x takes values in Ω_x and has average $\partial_j f(x)$.

Note: Such loops only exist if $\partial_j f(x)$ is in the convex hull of Ω_x.

Convex integration (2)

We want that $\partial_j f(x)$ lives in some open subset $\Omega_x \subseteq \mathbb{R}^3$.
We have a family of loops $\gamma : \mathbb{R}^2 \times \mathbb{S}^1 \to \mathbb{R}^3$ such that γ_x takes values in Ω_x and has average $\partial_j f(x)$.

Convex integration (2)

We want that $\partial_j f(x)$ lives in some open subset $\Omega_x \subseteq \mathbb{R}^3$.
We have a family of loops $\gamma : \mathbb{R}^2 \times \mathbb{S}^1 \to \mathbb{R}^3$ such that γ_x takes values in Ω_x and has average $\partial_j f(x)$.

Now let $N \gg 0$ and replace f by

$$g : x \mapsto f(x) + \frac{1}{N} \int_0^{Nx_j} [\gamma_x(s) - \partial_j f(x)] ds.$$

By a simple computation of partial derivatives, we see that

- $\partial_j g(x) \approx \gamma_x(Nx_j) \in \Omega_x$;
- $\partial_i g(x) \approx \partial_i f(x)$ for $i \neq j$;
- $g(x) \approx f(x)$.

The h-principle

We use this in much greater generality to formalize a seminal result in differential topology, Gromov's original h-principle (or homotopy principle).[1]

The homotopy principle provides a very general technique to construct solutions to partial differential relations.

Originally proven by Mikhael Gromov in 1973; we followed a proof by Mélanie Theillière from 2018.

[1] More precisely: the relative, parametric C^0-dense h-principle for open and ample first order differential relations on functions between smooth manifolds.

Formalization of the h-principle

We wanted to show that we can formalize deep geometric arguments in an interactive theorem prover.

We wrote a blueprint with a detailed LaTeX proof.

Formalization of the h-principle

We wanted to show that we can formalize deep geometric arguments in an interactive theorem prover.

We wrote a blueprint with a detailed LaTeX proof.

We had to formalize convex integration and jet spaces as part of the formalization.

We made 140 contributions to mathlib in the process, about convexity of sets, parametric integrals, differential geometry and various other topics.

This project took us about a year (part-time).

Final Result

```
theorem Smale :
  ∃ f : ℝ → 𝕊² → ℝ³,
    smooth (↿f : ℝ × 𝕊² → ℝ³) ∧
    f 0 = (coe : 𝕊² → ℝ³) ∧
    f 1 = (-coe : 𝕊² → ℝ³) ∧
    ∀ t, immersion (f t)
```

Talk Topics

In this talk:

1. Lean and mathlib
2. The sphere eversion project
3. A convenient formalization of vector bundles

Definition of a vector bundle

A bundle $\mathbf{E}$ over a base space B is just a dependent type $\mathbf{E} : B \to \text{Type}$.
It has total space $E := \Sigma_{x:B}\mathbf{E}(x)$ and projection function $\pi : E \to B$.
Example: the trivial bundle $\mathbf{E}(x) := F$.

Definition of a vector bundle

A bundle $\mathbf{E}$ over a base space B is just a dependent type $\mathbf{E} : B \to \text{Type}$.
It has total space $E := \Sigma_{x:B}\mathbf{E}(x)$ and projection function $\pi : E \to B$.
Example: the trivial bundle $\mathbf{E}(x) := F$.

A fiber bundle with fiber F is a bundle that is locally trivial.
This means that every $x_0 \in B$ has an open neighborhood U and a homeomorphism $\varphi : U \times F \xrightarrow{\sim} \pi^{-1}(U)$ that is a fiberwise map, i.e. $\pi(\varphi(x, y)) = x$ for all x, y. φ is called a local trivialization with base set U.

Definition of a vector bundle

A bundle $\mathbf{E}$ over a base space B is just a dependent type $\mathbf{E} : B \to \text{Type}$.
It has total space $E := \Sigma_{x:B}\mathbf{E}(x)$ and projection function $\pi : E \to B$.
Example: the trivial bundle $\mathbf{E}(x) := F$.

A fiber bundle with fiber F is a bundle that is locally trivial.
This means that every $x_0 \in B$ has an open neighborhood U and a homeomorphism $\varphi : U \times F \xrightarrow{\sim} \pi^{-1}(U)$ that is a fiberwise map, i.e. $\pi(\varphi(x, y)) = x$ for all x, y. φ is called a local trivialization with base set U.

A vector bundle is a fiber bundle where F and $\mathbf{E}(x)$ come with the structure of a vector space for each x, and the maps $y \mapsto \varphi(x, y)$ are fiberwise linear for each x.

Old definition in mathlib

```
class topological_vector_bundle : Prop :=
(locally_trivial : ∀ b : B,
  ∃ e : topological_vector_bundle.trivialization R F E,
  b ∈ e.base_set)
```

Old definition in mathlib

```
class topological_vector_bundle : Prop :=
(locally_trivial : ∀ b : B,
  ∃ e : topological_vector_bundle.trivialization R F E,
  b ∈ e.base_set)
```

Problems:

- separate definitions of trivializations for fiber and vector bundles;

Old definition in mathlib

```
class topological_vector_bundle : Prop :=
(locally_trivial : ∀ b : B,
  ∃ e : topological_vector_bundle.trivialization R F E,
  b ∈ e.base_set)
```

Problems:

- separate definitions of trivializations for fiber and vector bundles;
- definition is not standard for infinite-dimensional fibers.

Old definition in mathlib

```
class topological_vector_bundle : Prop :=
(locally_trivial : ∀ b : B,
  ∃ e : topological_vector_bundle.trivialization R F E,
  b ∈ e.base_set)
```

Problems:

- separate definitions of trivializations for fiber and vector bundles;
- definition is not standard for infinite-dimensional fibers.

Given trivialization φ defined on $U \times F$ and ψ defined on $V \times F$. We have a coordinate change function $C_{\varphi,\psi}$ that sends $x \in U \cap V$ to the continuous linear map

$$C_{\varphi,\psi}(x) := y \mapsto \psi^{-1}(\varphi(x, y)).2 : \mathsf{GL}(F).$$

We also need to require that if φ and ψ are trivializations of our vector bundle, then $C_{\varphi,\psi} : U \cap V \to \mathsf{GL}(F)$ is continuous.

New definition in mathlib

Together with Heather Macbeth I changed definition of fiber and vector bundles in Lean.

```
class fiber_bundle :=
(trivialization_atlas : set (trivialization F (π E)))
(trivialization_at : B → trivialization F (π E))
(mem_base_set_trivialization_at :
  ∀ b : B, b ∈ (trivialization_at b).base_set)
(trivialization_mem_atlas :
  ∀ b : B, trivialization_at b ∈ trivialization_atlas)
```

New definition in mathlib

Together with Heather Macbeth I changed definition of fiber and vector bundles in Lean.

```
class fiber_bundle :=
(trivialization_atlas : set (trivialization F (π E)))
(trivialization_at : B → trivialization F (π E))
(mem_base_set_trivialization_at :
  ∀ b : B, b ∈ (trivialization_at b).base_set)
(trivialization_mem_atlas :
  ∀ b : B, trivialization_at b ∈ trivialization_atlas)
```

A vector bundle is a fiber bundle that has fiberwise linear trivializations and continuous coordinate change

```
class vector_bundle [fiber_bundle F E] : Prop :=
(trivialization_linear :
  ∀ e ∈ trivialization_atlas F E, e.is_linear R)
(continuous_on_coord_change : ∀ (e e' ∈ trivialization_atlas F E),
  continuous_on (e.coord_changeL R e' : B → F →L[R] F)
  (e.base_set ∩ e'.base_set))
```

Concluding thoughts

- The new vector bundle definition deduplicates API and is convenient to work with.
  - Example: defining the tangent bundle of a smooth manifold takes 60 lines, pullbacks (3 definitions) are 120 lines.
- Type-classes are great.
  - We introduced a type-class stating that a trivialization `e` is part of the atlas of trivializations of the fiber bundle.
- Don't hesitate to refactor definitions to make them easier to use.

Thank You

The h-principle

Theorem (Gromov, 1973)

If $\mathcal{R}$ is an open and ample[2] partial differential relation for functions between manifolds then $\mathcal{R}$ satisfies the h-principle, i.e. any formal solution can be smoothly deformed into a holonomic one inside $\mathcal{R}$.

[2] Ampleness is a geometric condition that ensures that certain convex hulls are large enough for the convex integration argument to work.

Future plans

I want to formalize more algebraic topology, not necessarily using HoTT.

Future plans

I want to formalize more algebraic topology, not necessarily using HoTT.

An excellent target would be to compute various homotopy groups of spheres.

Computations of such algebraic invariants are involved and prone to error.

The Kenzo program is able to compute algebraic invariants of many complicated spaces.

I would like to write a formally verified algorithm that can automatically compute such algebraic invariants.

Kenzo

For example, a published 2010 paper mistakenly claims that

$$\pi_4(\Sigma K(A_4, 1)) = \mathbb{Z}/4\mathbb{Z}.$$

(On homotopy groups of the suspended classifying spaces, Theorem 5.3)
However, it was later found by that

$$\pi_4(\Sigma K(A_4, 1)) = \mathbb{Z}/12\mathbb{Z}$$

with the help of the Kenzo program.

What can we hope for in the future

- On-line textbooks where you can see proofs in as much or as little detail as you want;
- A digital referee, that will check the proofs in a paper submitted to a journal for correctness;
- Maybe even a mathematical assistant, that will work out details of proof sketches.