Schemes in Lean

Kevin Buzzard, Kenny Lau, Chris Hughes, Ramon Fernandez Mir

Imperial College London

AITP, 11th April 2019.

Overview of the talk

[**note: links in this pdf are clickable.**]

- A *scheme* (for the purposes of this talk) is a technical object in modern algebraic geometry.
- I'm an algebraic number theorist.
- I'm going to tell the story of how me and a team of mathematics undergraduates defined the notion of a scheme in Lean.
- Lean is an ITP, it uses dependent types and the calculus of inductive constructions.
- The weirdest part of the story is that none of us had a clue what an ITP was two years ago.

Lean

- I got interested in interactive theorem proving just under two years ago.
- Why? For reasons unrelated to this talk.
- I learnt some Haskell and I played around with Coq for two weeks.
- I am a formalist and found these new (to me) toys very appealing to use.
- Then I watched Tom Hales' talk in the 2017 Big Proof conference, and he mentioned Lean, so I thought I'd give Lean a go.

- I am interested in learning and interested in teaching.
- Summer 2017: I began to teach myself Lean.
- Learning an ITP from scratch with no background is hard.
- However there is an *incredibly* helpful, polite, responsive and useful live Lean chat at Zulip – leanprover.zulipchat.com/.
- Without this chat I would never have been able to start.
- October 2017: I began to teach my "introduction to proof" course at Imperial College.
- I also started doing the problem sheets from *my own course* in Lean.
- I very quickly realised how difficult this formal proof verification malarkey was.

Introduction

The beginnings.

The project starts.

What is a scheme?

Simple type theory / FOL rant slide

First success.

Equality and cheating.

Schemes 2.0

The future.

First part of the first question on the first problem sheet:
Q1) True or false: $x^2 - 3x + 2 = 0 \implies x = 1$.

Me) "False: set $x = 2$".

Lean) OK so now you have to prove $2^2 - 3 \times 2 + 2 = 0$ and that $2 \neq 1$.

Me) . . .

Undaunted by the facts that
(a) I could not prove the first question on the first problem sheet, and
(b) the vast majority of my colleagues had no interest in, and no understanding of what I was doing,
I did three more things:

- I started a blog – xenaproject.wordpress.com. The spin: Xena is a "virtual undergraduate" taking my course. Can she (i.e. can I) do the problem sheet questions? Can she pass the exam?
- I got an account at github and started a completely chaotic "xena" repository.
- I started a club – the Xena project club. Meets on Thursday evening in the computer room at Imperial College maths department.
- It was the blind leading the blind, at first.
- And then some people started bubbling to the top.

- 5th November 2017 : Kenny Lau (1st year Imperial maths UG) had a PR accepted into Lean's maths library. Subspaces of vector spaces; linear maps, and kernels etc.
- Six weeks earlier, Kenny Lau had never heard of theorem provers.
- My response: I immediately got Kenny to start doing commutative ring theory in Lean. (note to self: write vague intro to rings on the blackboard)
- January 2018: I had my first PR accepted – the complex numbers! (an indication of the state of Lean's maths library in early 2018).
- Feb 2018 Chris Hughes (another 1st year Imperial maths UG) had his first PR accepted.
- Fast forward to now – Chris has had 162 commits accepted and is now a maintainer. He has proved quadratic reciprocity, Sylow's Theorems, the fundamental theorem of algebra,. . . all in Lean.

By Feb 2018 Kenny Lau had written a robust API for localization of rings. “localization” means “adding a controlled amount of division”.

I proposed to Kenny that we should collaborate on defining a scheme in Lean. Our source was Johan de Jong’s *stacks project* stacks.math.columbia.edu/

Introduction

The beginnings.

The project starts.

What is a scheme?

Simple type theory / FOL rant slide

First success.

Equality and cheating.

Schemes 2.0

The future.

Schemes – a bridge between geometry and algebra.

Schemes were invented by Alexandre Grothendieck around 1960, to put algebraic geometry back on a rigorous footing.

They are a brilliant example of a mathematical *definition*.

(board explanation of affine schemes)

Introduction

The beginnings.

The project starts.

What is a scheme?

Simple type theory / FOL rant slide

First success.

Equality and cheating.

Schemes 2.0

The future.

$\mathbb{C}[X]$ is the *ring of polynomials in one variable.*

Example of an element: $f(X) = 3X^2 + \pi X + \sqrt{-1}$.

You can think of $f(X)$ as a function $\mathbb{C} \to \mathbb{C}$.

What about $1/X$ though? This is not a polynomial.
However it is still a *function* – not on all of $\mathbb{C}$, but on $\mathbb{C} - \{0\}$.

Localization of rings is the theory of adding "controlled denominators" to rings. This is what Kenny had made, and this is what Grothendieck needed to define functions on his schemes.

Technical interlude (lasts for one slide only). A scheme is:

- A topological space X;
- a ring (the "ring of functions") associated to each open subset of X;
- a bunch of axioms (X is a locally ringed space and locally affine).

Discussion point: Attaching a ring to every open subset of a set or type seems to me to be *hard to do* in HOL light, Isabelle-HOL, etc., and *impossible to do* in first order logic[1]. FOL / Simple type theory is not well suited to MSc/beginning PhD level geometry. For me, as an arithmetic geometer, this is a *big turn-off*. This sheaf (the ring attached to each open set) is a *dependent type* which shows up naturally in what mathematicians regard as "basic mathematics".

[1] It was pointed out to me in the talk that of course you can just add the axioms of ZFC and do it there.

Introduction

The beginnings.

The project starts.

What is a scheme?

Simple type theory / FOL rant slide

First success.

Equality and cheating.

Schemes 2.0

The future.

- Kenny and I managed to define schemes in Lean by March 2018.
- I then started asking about comparing our work with formalisations in other theorem provers. It turned out that as far as I could see, there were no other formalisations of a scheme in a theorem prover.
- Discussion point: 99% of mathematicians are not going to be interested in formal proof verification systems until they can see the kind of objects that they think about formalised in these provers. And many mathematicians think about schemes.

Introduction

The beginnings.

The project starts.

What is a scheme?

Simple type theory / FOL rant slide

First success.

Equality and cheating.

Schemes 2.0

The future.

Our announcement of the formal definition of a scheme led to enthusiastic discussion on the Zulip chat.
Q) “Did you formalise any examples?”
Me) No...
Q) “Well how do you know you have not made an error?”
Me) . . .

I told the chat that it would probably not be too difficult to prove that an affine scheme was a scheme. I was told that this was an important part of the process. So off we went again.

Introduction

The beginnings.

The project starts.

What is a scheme?

Simple type theory / FOL rant slide

First success.

Equality and cheating.

Schemes 2.0

The future.

Let me spend a few minutes explaining the biggest headache that we had when defining schemes and examples of schemes.

If we take the ring $\mathbb{C}[X]$ and invert X^2, we get the polynomial functions defined on the subset of $\mathbb{C}$ where $X^2 \neq 0$. Which is $\mathbb{C} - \{0\}$.

If we take the ring $\mathbb{C}[X]$ and invert X^3, we get the polynomial functions defined on the subset of $\mathbb{C}$ where $X^3 \neq 0$. Which is $\mathbb{C} - \{0\}$.

So what *is* the ring of polynomial functions on $\mathbb{C} - \{0\}$? Is it $\mathbb{C}[X][1/X]$ or $\mathbb{C}[X][1/X^2]$ or $\mathbb{C}[X][1/X^3]$? A mathematician does not care, because all of these rings are *equal*.

From the point of view of type theory however, these rings are not equal – they are merely *canonically isomorphic*.

Canonical isomorphisms.

Wikipedia (Isomorphism page): “A canonical isomorphism is a canonical map that is an isomorphism.”

Wikipedia (Canonical Map page): “In mathematics, a canonical map, also called a natural map, is a map or morphism between objects that arises naturally from the definition or the construction of the objects being mapped against each other. In general it is the map which preserves the widest amount of structure, and it tends to be unique. In the rare cases where latitude in choice remains, the map is either conventionally agreed upon to be the most useful for further analysis, or sometimes simply the most elegant or beautiful known. “

Me) . . .

I have even used the word “canonical” in my papers! It has *no formal definition*!

The mathematician's super-power.

Quote from "Etale cohomology" (a book about schemes!) by J. S. Milne.
"Notation: a canonical isomorphism is denoted by $=$."

By "$=$" here we mean "mathematical equality". Mathematicians can rewrite using it! It is an extremely powerful tool – more powerful than any tool in any theorem prover. It enables us to *cheat*.

Lean would not let me cheat, so I had to work.

- We needed some technical lemmas in ring theory.
- Enter Chris Hughes, who proved some technical combinatorial lemmas.
- After he did what I asked him to do, I ran into the issue that I had lost my super-power of being able to rewrite along canonical isomorphisms.
- I wanted to apply Chris' lemmas to rings which were canonically isomorphic to the ones he'd used.
- Lean at that time had no tactic for doing this.
- I did it by hand myself. The code was unreadable.

Introduction

The beginnings.

The project starts.

What is a scheme?

Simple type theory / FOL rant slide

First success.

Equality and cheating.

Schemes 2.0

The future.

- May 2018: affine schemes were schemes.
- And there *was* a mistake in our original definition :-)
- And the code base had been written by three complete amateurs.
- What now?
- Enter Ramon Fernandez Mir (final year maths/computing MSc student in 2018–19).
- Ramon's MSc project: start again from scratch and do schemes again. And this time do them *right*.

- Wednesday 10th April 2019: Ramon finished defining schemes, affine schemes, and locally ringed spaces.
- github.com/ramonfmir/lean-scheme
- The work also incorporates a new technical innovation due to Neil Strickland.
- Strickland defined an easy-to-check predicate $P(R, D, A)$ saying "A is *isomorphic* to the ring $R[1/D]$ you get if you start with R and then allow division by everything in D".
- Much of the ringy part of Chris' code was completely rewritten using this predicate.

Conclusions.

- It costs too much money to formalise detailed proofs of hard theorems.
- It is not clear to me that mathematicians would be drawn in by this anyway.
- It is much cheaper to formalise technical *mainstream* mathematics PhD level definitions.
- I have evidence that it is possible to do this in Lean.
- But essentially nobody is doing this. Why is this?

- In 2011 Peter Scholze came up with a far far more complex definition – a *perfectoid space*.
- He used them to make some serious progress on "the Langlands philosophy".
- Scholze was awarded the Fields Medal in 2018.
- I'm defining a perfectoid space in Lean, with Patrick Massot (Orsay) and Johan Commelin (Freiburg). Both mathematicians!
- I had hoped to announce the completion of the project today, but we are three relatively straightforward arguments away from the definition.
- Apologies to Michael Kinyon for keeping him awake until the small hours every night so far!
- Final (discussion?) point: My impression is that proving even the simplest statements about perfectoid spaces is completely out of reach of an ATP. I believe this will still be the case in ten years' time. I would *love* to be proved wrong.