Making and using mathematical objects in dependent type theory

Kevin Buzzard

Imperial College London

29th May 2019.

I want to digitise some parts of mathematics. Which parts?

- Example: Imperial College London's new undergraduate mathematics curriculum (the Xena project).
- Example: Modern complex mathematical objects, and statements of modern deep mathematical theorems and conjectures (Hales' FABSTRACTS).
- I reserve the right to come up with more examples later.

Why do I want to do this?

- Because it's fun.
- Because the projects suggested above are *manifestly accessible*.
- (Xena): because there might be consequences for mathematics education (Thoma / Iannone).
- (FABSTRACTS): Because there might be consequences for search / AI.
- Unshaking belief that the approach above is more likely to bring "normal" mathematicians on board than other, more "serious", formalization projects.
- I reserve the right to come up with more consequences later.

This talk is about the pain points which “normal mathematicians” run into when trying to formalise “normal mathematics” in the Lean theorem prover (dependent type theory, quotient axiom, all proofs are equal, undecidable definitional equality).

The three main formalisation projects I have worked on in my two years in this area:

- Formalising solutions to my undergraduate course M1F – the “introduction to proof” course in the maths department at Imperial College London.
- Formalising Grothendieck’s definition of a scheme (with Kenny Lau, Chris Hughes, Ramon Fernandez Mir).
- Formalising Scholze’s definition of a perfectoid space (with Patrick Massot, Johan Commelin).

I also supervised 20 Lean student projects last summer, and watched carefully to see where students struggled.

I believe that these projects have each played a role in achieving my ultimate goal of driving development of Lean's maths library *in the direction that I want it to go*.

That direction is: directly into the face of a modern pure mathematician (who works in a maths department and is doing number theory, geometry, analysis, algebra or topology – not "foundational" mathematics).

I am not naive enough to believe that Lean is "the one true solution" to getting mathematicians involved in computer proof. However I do believe that it is *essential* that more mathematicians get on board. FABSTRACTS is *manifestly* a feasible project, but I would like more people involved, and I think that some of them have to be mathematicians. And preferably mathematicians with some experience in "traditional" mathematics.

The pain points that mathematicians run into when formalising in Lean, are typically in the following four categories.

- Type 1: Basic learning curve issues with the software.
- Type 2: Missing tactics ("Coq can do it but Lean can't").
- Type 3: Consequences of not understanding the algorithms being used by the software.
- Type 4: The frustration of formalising ideas which are "obvious to a mathematician".

Learning curve issues: for me the problem was that the documentation was written "by computer scientists, for computer scientists". Solution which I will implement this summer: write my own documentation – Lean for mathematicians. Follow the Xena project wordpress blog and/or follow me on Twitter `@XenaProject` for updates.

The rest of this talk is examples.

Example: What is a group?

It is:

- A set G;
- A function $m : G \times G \to G$;
- A function $i : G \to G$ and an element $e \in G$ which you can work out from m;
- some axioms which m and i and e must satisfy.

Question: How is this definition "packaged up" in, say, ZFC set theory? Is a group an ordered pair (G, m), or an ordered 4-tuple (G, m, i, e), or an ordered 5-tuple $(\texttt{group}, G, m, i, e)$ or what?

Answer (all normal mathematicians in unison): "Who cares!"

If you're making groups, or more complicated things like perfectoid spaces, in Lean, then *you have to learn to care*. At least to a certain extent.

Sometimes the way to figure out the best way to do something is to do it in an arbitrary way and then see what happens.

Asking the CS community "what's the best way to do this?" and getting the response "Try this way and see if it works OK" was sometimes frustrating.

Status of this issue: I have learnt that sometimes this game is an experimental science.

Example (1st part of first question on first example sheet in my 1st term 1st year course).

True or false? If $x \in \mathbb{R}$ then $x^2 - 3x + 2 = 0 \implies x = 1$.

Mathematician's solution: False: set $x = 2$.

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

Mathematicians don't know what to do next. And "Theorem Proving In Lean" does not tell them.

Status of this issue: **solved.** `by norm_num`.

Mathematicians and computer scientists working together. There's an analogous story with `ring`, a tactic for proving $(a + b)^3 = a^3 + 3a^2b + 3ab^2 + b^3$. If students learnt stuff in school, they want to use it.

Another example which I found shocking, and which undergraduates find painful, was this issue that $\mathbb{N} \not\subset \mathbb{Z}$ and so on.

This caused pain for undergraduates formalising basic number theory.

Status: this might have been solved in the last few weeks. Some new tactics just appeared, written by Paul-Nicolas Madelaine. I have yet to try them.

Example (still on 1st example sheet): let $X = \{1, 2, \{2\}, \{1, 2\}\}$. Is $2 \in X$? Is $\{2\} \in X$? Is $\{2\} \subseteq X$? etc. Bonus question: is $2 \subseteq X$? Does this even make sense?

Why do I care? Because I want to teach students what $\in$ and $\subseteq$ mean.

It was a shock to me to find that types, which had been sold to me as "just like sets", were not a good fit for this question.

The point of the first example sheet is to test the students' understanding of basic notation such as $\implies$, $\in$ and $\subseteq$. This pathological question is an exercise in logic.

Status: reluctant realisation that it is difficult to ask this sort of question in type theory.

This solution is unsatisfying to me, because mathematics undergraduates are prone to misusing the notation $\{\}$, confusing $S \cup T$ with $\{S, T\}$ for example.

Example: Euler's formula $F - E + V = 2$.

This is part of my introduction to proof course. I always hated teaching this part of the course, but never really figured out why. Now I know why. It is because the treatment of this formula in the book for the course is *extremely informal* and very difficult to formalise as it stands.

"Now imagine that your polyhedron is made out of glass, and put your eye very close to a face. You will see a planar graph". It is not hard to come up with explicit counterexamples to reasonable interpretations of this statement.

Status: I decided to reject the proof in the book, and instead railed against the lack of rigour.

Example: Here's some mathematics from my 1st year exam last year.

Question. Let S be a set with two elements, and let $\star$ be a reflexive binary relation on S. Prove that S is transitive.

Solution. WLOG $S = \{0, 1\}$. **Hold it right there.**

Unwritten lemma: Say S and T are sets, and $f : S \to T$ is a bijection. Say $\star$ is a binary relation on S, and $f(\star)$ is the obvious induced binary relation on T. If $\star$ is reflexive/transitive, then $f(\star)$ is reflexive/transitive. A mathematician does not need to explicitly say this! S and T are "the same" and the result is *obvious to them*. And yet it's there.

I believe that this proof should be automatically generated by a `transport` or `transfer` tactic. That tactic does not yet exist, and I am unclear why. I think that it is because I am demanding that its scope is *gigantic*.

Example: Say R and S are commutative rings, and R is isomorphic to S. Say R is Noetherian, local, Cohen-Macauley, and has finite global dimension. Are these properties also true for S?

Ask any mathematician this question, *even one who has no idea what these definitions mean*. They are very likely to guess "yes". And if they know what the definitions mean, they will even helpfully tell you that the assertion is "obvious".

This is because of the **mathematicians' code of conduct.**

- Rule 1: Do not make definitions which are not isomorphism-invariant.
- Rule 2: It is OK to use your super-powers to verify that rule 1 is being adhered to; no proof necessary.

The mathematician's super-power is that they have a *profound and intrinsic grasp* of what it means for two objects to be *the same*.

Being the same.

Whether or not two objects are *the same* depends on context!

For example, two subgroups of a group might be (a) equal, (b) conjugate or (c) isomorphic.

Whether or not they are “the same” depends on what we are doing with them, and we even reserve the right to change our definition of “the same” depending on what we are doing.

Status: computer scientists are finding the `transfer` tactic hard to write, and at least one of the reasons is that mathematicians have thus far been very imprecise about the scope of the tactic. The general idea: if `P : Group -> Prop` is a “mathematical predicate” then `P` should be constant on equivalence classes, where two groups are equivalent if they are “the same”.

As Patrick Massot will tell you, I am very bad at writing “road maps” for formalisation projects. I like to dive in and see where we end up.

A fabulous example of this came up when formalising schemes.

To explain this example, I need to talk about localisations of rings. So here is ten minutes of pure mathematics on the board.

Even in EGA, we find usage of $=$ in this context, with Grothendieck very happy to write that $R[1/r] = R[1/r^2]$ and $R[1/f][1/g] = R[1/fg]$. I believe that this is because mathematicans have a clear mental model of the process of localisation, mostly informed by the situation of $\mathbb{Z} \subseteq \mathbb{Q}$.

What is actually going on is that $R[1/r]$ satisfies a *universal property*.

Mathematicans have a very strong notion of "the same" here. If two objects both satisfy the same universal property, then they are uniquely isomorphic. Hence they are regarded by most mathematicians as being *the same object*.

The problem: even if $R[1/r]$ and $R[1/r^2]$ are "the same", they are not *equal*, in any sense, in Lean's dependent type theory.

This issue multiplies and multiplies.

I foolishly asked Chris Hughes to prove a lemma in the stacks project relating the rings R, $R[1/r]$ and $R[1/rs]$ for various choices of R and $r, s \in R$.

Chris proved *exactly* the lemma as stated in the stacks project.

The problem was that this lemma was *not at all* equal to the lemma we needed. It was just "the same" as it.

The *actual* lemma we needed was when $R = A[1/q]$, and instead of $R[1/r]$ we needed to use $A[1/qr]$ – a ring which was "the same" as $A[1/q][1/r]$. Even the r's are "the same" but not equal.

There was a whole bunch of diagrams which we needed to check commuted, via some uniqueness arguments, some of which were quite delicate. I wrote a bunch of spaghetti code.

It got to the point where I was convinced that Grothendieck's approach was not rigorous; there seemed to be so many checks which had not been undertaken.

My reaction at the time was that this whole mess should be solved by a `transfer` tactic. But when pressed, I found that I was not even clear about what this tactic should do.

We had theorems about images of maps between rings, and we actually wanted theorems about images of maps between other rings which were “the same”.

I am still a bit confused about how a tactic could help me here.

Possible solutions to this mess.

- (1) Reduce to the universal case, where things are integral domains.
- ...this sort-of relies on a coincidence.
- (2) Don't prove anything about $R[1/r]$ and instead prove results about any ring satisfying the universal property $R[1/r]$ satisfies.
- Problem: possible universe issues, have to re-interpret images and kernels via functors.

Example of something we'd need with method (2): Kenny Lau had proved that the elements of R which get mapped to zero in $R \to R[1/r]$ are precisely the $a \in R$ such that there exists some n with $ar^n = 0$ in R. We'd need this for any map $R \to A$ satisfying the universal property.

And then Neil Strickland made a decisive contribution.

Neil wrote down a small list of properties that a map $R \to A$ had to satisfy which were necessary and sufficient for A to be isomorphic to $R[1/r]$. No quantifying over all rings, just a neat little bit of algebra.

The issue now became: to prove the analogue of Chris' lemma not for $R[1/r]$ but for all rings satisfying these properties.

Miraculously, the only things Chris needed about $R[1/r]$ in his proof was that it satisfied precisely the properties that Neil had isolated.

Status of this issue: solved, although I don't know whether we got lucky. We were forced to work harder than expected, because we lost our super-powers.

Example: What is a perfectoid space?

Johan Commelin, Patrick Massot and myself implemented the definition in Lean, which involved well over 10,000 lines of code.

We reluctantly had to start caring about implementation issues.

And hence we reluctantly had to start learning about them.

Issues we faced:

- Total number of people on the planet who could help seemed small at times.
- “Here is 10,000 lines of code, why does it time out on line 9000?” is not an appealing question.
- The issues that arose were *not mathematics*. “This `rfl` should work in theory”.

What we learnt:

- Don’t brandish big objects around.
- An object can be bigger than you think.
- Knowing something about Lean’s unification algorithms can definitely help.
- Debugging is tedious, especially when it’s not mathematics.

So where now?

The one issue which I'm sure will bite me again is this concept of two types being "the same".

Strickland pulled us out of the fire once, but our luck might run out some day.

When we do non-trivial examples of perfectoid spaces, we will need some topological ring analogy of the trick.

But *every other issue I have run into so far has been solved*.

Final example of an issue we (mostly Floris van Doorn) solved this week:

```
/- A cube cannot be cubed. -/
theorem cannot_cube_a_cube :
  ∀{n : ℕ}, n ≥ 3 →                                  -- In ℝ^n for n ≥ 3
  ∀{ι : Type} [fintype ι] {cs : ι → cube n},         -- given a finite collection of (hyper)cubes
  2 ≤ cardinal.mk ι →                                -- containing at least two elements
  pairwise (disjoint on (cube.to_set ∘ cs)) →        -- which is pairwise disjoint
  (⋃(i : ι), (cs i).to_set) = unit_cube.to_set →     -- whose union is the unit cube
  injective (cube.w ∘ cs) →                          -- such that the widths of all cubes are different
  false :=                                           -- then we can derive a contradiction
begin
  intros n hn ι hι cs h1 h2 h3 h4, resetI,
  cases n, rw [ge, ←not_lt] at hn, apply hn, norm_num,
  exact not_correct ⟨h2, h3, h4, h1, hn⟩
end
```

Conclusions:

- Lean seems to be up to the task that Hales has set for it with his FABSTRACTS project.
- Mathematicians *can* learn to use it.
- There was enough support from the CS community to solve pretty much everything so far.
- Mathematicians are coming up with new ideas which will hopefully make this whole area seem more interesting to “regular mathematicians”.