Teaching computers to prove theorems

K. Buzzard, Imperial College London

10th May 2021, Combinatorics on words

Before we start

Two things before we start:

1) Thank you to Narad and Manon for the invitation, and thanks to you all for coming!

2) I am an algebraic number theorist (not a computer scientist)

Computers and computation

Traditional use of computers in mathematics: to *compute*.

Computer algebra packages:

- Mathematica
- Maple
- Sage
- Pari-gp
- Magma
- Gap
- Matlab
- ...

You have probably heard of (and maybe used) some of these.

Computers and proving

Lesser known use : to *reason*.

Computer proof packages:

- Lean
- Coq
- Isabelle/HOL
- Metamath
- Mizar
- HOL 4
- HOL Light
- Agda
- ...

You have probably not used, or even heard of, these systems. Let's compare and contrast. [cut to pari/Lean]

University teaching

Do universities teach students how to use computer proof systems?

Sure! In the computer science department.

At my university, we have now started to teach *maths* students how to use computer proof systems.

Some of the world experts in doing mathematics in the Lean theorem prover are mathematics undergraduates at my university.

The "big three"

Lean is a free and open source computer proof system, written by Leo de Moura at Microsoft Research.

Coq is a free and open source computer proof system which has existed for over 30 years.

Isabelle/HOL is a free and open source computer proof system and Cambridge UK is one of the hubs for this system.

Why do computer scientists keep writing computer proof systems if mathematicians do not use them?

It's kind of a fluke

These systems are used (in computer science departments, and in industry) to *find bugs in computer code.*

Pentium fdiv bug cost Intel half a billion dollars.

The systems can be used to prove that computer code functions correctly.

The same systems can be used to prove theorems in pure mathematics.

Lean in 2017

Me in 2017: “Can I use Lean to help students to learn my introduction to proof course?”

Sheet 1, Q1, Part (a). “True or false: If x is a real number, then $x^2 - 3x + 2 = 0 \implies x = 1$.”

My model answer: “False: set $x = 2$. Type it into Lean.

Lean says that it now suffices to prove two things:

- $2^2 - 3 \times 2 + 2 = 0$;
- $2 \neq 1$.

I realised I was in love. I was also stuck.

A week later, an MSc student (Mario Carneiro) in the philosophy department at Carnegie Mellon wrote a Lean “tactic” which solved both of these goals.

The other problem sheets

Sheet 3 had complex numbers.

At that time, Lean had no complex numbers.

So I defined a complex number to be a pair of real numbers, and proved they were a ring.

$e^{i\theta} = \cos(\theta) + i\sin\theta$: needed a theory of convergence of complex power series. Oops. I got 1st year undergraduate Chris Hughes to work on it.

Basic number theory and congruences: much easier.

Equivalence relations: *really easy.*

Student: “I did not understand equivalence relations, and then I tried doing them in Lean, and now I understand them”.

Schemes

Some undergraduates started getting really good.

I told Kenny Lau (1st year UG) to try formalising the theory of localisation of commutative rings in Lean.

Two weeks later he was done.

I suggested we work together to formalise the definition of a scheme.

By March 2018 Lau, Hughes and I had a definition and were proving lemmas about schemes.

The penny dropping

I then started to ask how this had been done in the other theorem provers.

I was told that it had not been done before, and hence our work was probably publishable.

Two questions raised by this story:

1. Can we use systems like this to teach graduate students (or even undergraduates) about schemes?
2. If these systems can handle schemes, how much more can they handle?

A new way of doing mathematics

By 2019 Commelin, Massot and myself had proved that they could handle the definition of a perfectoid space.

These systems have been being used by computer science professors for *decades*.

Their tastes in mathematics do not always coincide with ours.

The two questions I will discuss in the last four slides of the talk:

- How can we get mathematicians to use this software?
- *Why* should mathematicians use this software?

How?

Two obvious approaches:

1) Teach young people to do BSc and MSc level mathematics in these systems.

2) Do modern research-level work in these systems.

Combinatorics, and Words

Bhavik Mehta is a PhD student in Cambridge.

In 2019 I encouraged him to work on the Kruskal Katona theorem.

Since then: major contributions in category theory, and now doing convex geometry with an undergraduate.

Chris Hughes is an MSc student at Imperial College.

His MSc thesis: several tactics to solve the word problem in finitely presented groups.

Formalising work of Scholze

Peter Scholze approached the formalisation community and challenged them to check one of his recent theorems with Clausen.

A team of people are working on this, and we will succeed.

The point is not the theorem – the point is what it tells us about library development.

Why?

- Gamifying undergraduate level mathematics makes for better teaching.
- Computers marking homework.
- Interactive error-free undergraduate textbooks.
- Searchable databases of PhD level theorem statements (Stacks project).
- A fully formalised pure mathematics undergraduate degree.
- Computers knocking off tedious lemmas for researchers – a “digital graduate student”.
- AI experiments and the beginning of a new era.

Thank you all for coming.

#ads

- Play the natural number game!
- Check out my `xenaproject.wordpress.com` blog for stuff accessible to mathematicians.
- Try to formalise the *statement* of a theorem you're interested in, in Lean, and ask on the 24/7 helpline `leanprover.zulipchat.com` if you get stuck.
- Tell your smart mathematics undergraduates about Lean and send them to the Xena Project discord server.
- `leanprover-community.github.io` is the community website.