Will computers prove theorems?

Kevin Buzzard, Imperial College London

ICMS-LMS undergraduate series 2022-23, 29th March 2023

Before we start

Hello! I’m Kevin.

Thanks a *lot* to the LMS/ICMS for this invitation.

I won’t be looking at the chat *during* the talk, but I will be around for questions afterwards.

What’s this talk about?

I will talk about

- The strengths and weaknesses of large language models when it comes to mathematics;
- The strengths and weakness of interactive theorem provers when it comes to mathematics;
- Where we are, and where we’re going.
- What it’s like to do research in the area of interactive theorem provers.

Computers in mathematics

In the “old days”, computers were used by mathematicians to *compute*!

Computing can give *evidence* towards a conjecture.

Example: Goldbach’s conjecture says that every even $n \geq 4$ is the sum of two primes.

Verified for $4 \leq n \leq 4 \times 10^{18}$ by computer.

Obviously such an approach can’t *prove* the Goldbach conjecture.

Computers in mathematics

Let’s take a simpler problem: prove that there are infinitely many prime numbers!

We can write a python program which starts printing them out, and observe that it doesn’t seem to stop.

Python can be used to *compute*, but it cannot be used to *reason*.

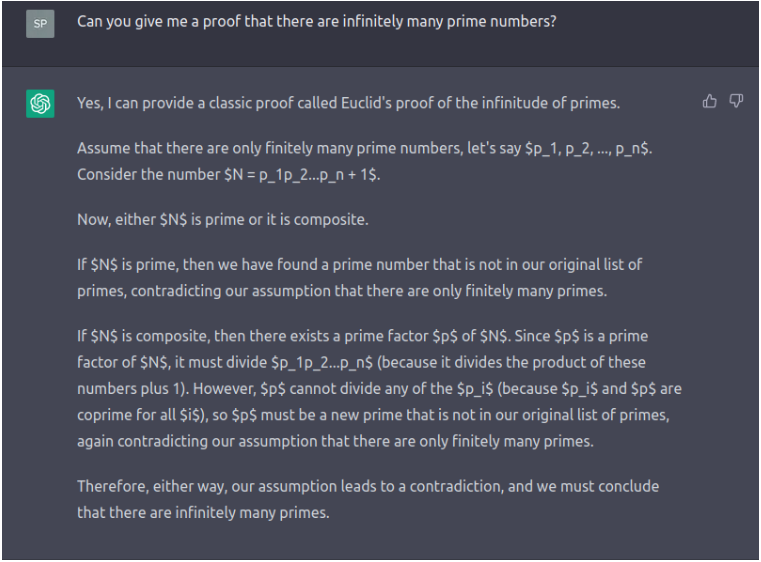
But nowadays computers can be used in new ways, and we *can* use them to prove that there are infinitely many primes.

Computers in mathematics

Let’s use a “large language model” (LLM) to prove that there are infinitely many prime numbers.

I’ll use ChatGPT.

ChatGPT



Can you give me a proof that there are infinitely many prime numbers?

Yes, I can provide a classic proof called Euclid's proof of the infinitude of primes.

Assume that there are only finitely many prime numbers, let's say $p_1, p_2, ..., p_n$. Consider the number $N = p_1p_2...p_n + 1$.

Now, either N is prime or it is composite.

If N is prime, then we have found a prime number that is not in our original list of primes, contradicting our assumption that there are only finitely many primes.

If N is composite, then there exists a prime factor p of N. Since p is a prime factor of N, it must divide $p_1p_2...p_n$ (because it divides the product of these numbers plus 1). However, p cannot divide any of the p_i (because p_i and p are coprime for all i), so p must be a new prime that is not in our original list of primes, again contradicting our assumption that there are only finitely many primes.

Therefore, either way, our assumption leads to a contradiction, and we must conclude that there are infinitely many primes.

Lean

Now let’s use an interactive theorem prover (ITP) to prove that there are infinitely many prime numbers.

I’ll use Lean.

Cheating

So what just happened?

There are a lot of versions of Euclid's proof that there are infinitely many primes on the internet.

ChatGPT is trained on the internet.

So it just “followed a human proof”, in some sense.

Cheating

The Lean proof invoked a “tactic”, which searched Lean’s mathematics library `mathlib` for a proof.

That proof was written by a human.

More precisely, the proof (a variant of Euclid’s) was *translated* from human languge to Lean’s language, by a human.

Conclusion

At no point is a machine doing anything approaching “thinking by itself”.

However, at the end of the day, we do have computers proving infinitude of primes, via two totally different methods.

A harder question

So let’s move onto a harder question.

There are infinitely many primes which end in 7.

This is because 7 and 10 are coprime, and Dirichlet proved that if a and b are coprime positive integers then there are infinitely many primes congruent to a mod b.

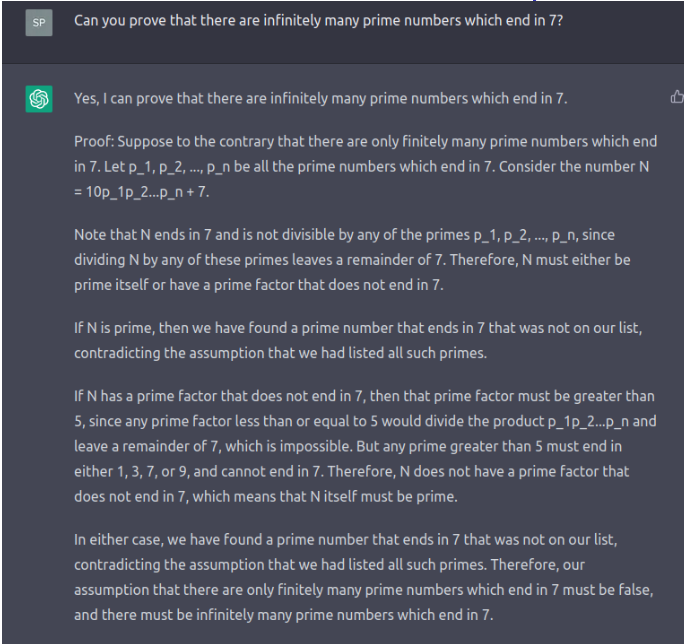
I don’t know of an elementary proof though.

The problem with Euclid’s strategy is that a large number which ends in 7 can be the product of a prime ending in 3 and a prime ending in 9 (example: 57).

A harder question

Let's see what the LLM makes of this.

A harder question



Can you prove that there are infinitely many prime numbers which end in 7?

Yes, I can prove that there are infinitely many prime numbers which end in 7.

Proof: Suppose to the contrary that there are only finitely many prime numbers which end in 7. Let p_1, p_2, ..., p_n be all the prime numbers which end in 7. Consider the number N = 10p_1p_2...p_n + 7.

Note that N ends in 7 and is not divisible by any of the primes p_1, p_2, ..., p_n, since dividing N by any of these primes leaves a remainder of 7. Therefore, N must either be prime itself or have a prime factor that does not end in 7.

If N is prime, then we have found a prime number that ends in 7 that was not on our list, contradicting the assumption that we had listed all such primes.

If N has a prime factor that does not end in 7, then that prime factor must be greater than 5, since any prime factor less than or equal to 5 would divide the product p_1p_2...p_n and leave a remainder of 7, which is impossible. But any prime greater than 5 must end in either 1, 3, 7, or 9, and cannot end in 7. Therefore, N does not have a prime factor that does not end in 7, which means that N itself must be prime.

In either case, we have found a prime number that ends in 7 that was not on our list, contradicting the assumption that we had listed all such primes. Therefore, our assumption that there are only finitely many prime numbers which end in 7 must be false, and there must be infinitely many prime numbers which end in 7.

A harder question

Nice try, but very wrong.

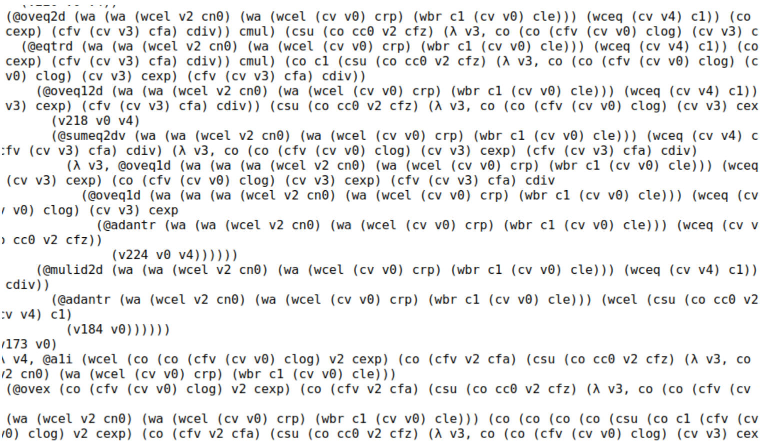
It’s what an undergraduate might write – which is not unexpected.

So how does the theorem prover fare on this harder question?

Well, it can do it!

Let me show you part of the proof.

A harder question



```
(@oveq2d (wa (wa (wcel v2 cn0) (wa (wcel (cv v0) crp) (wbr c1 (cv v0) cle))) (wceq (cv v4) c1)) (co
cexp) (cfv (cv v3) cfa) cdiv)) cmul) (csu (co cc0 v2 cfz) (λ v3, co (co (cfv (cv v0) clog) (cv v3) c
 (@eqtrd (wa (wa (wcel v2 cn0) (wa (wcel (cv v0) crp) (wbr c1 (cv v0) cle))) (wceq (cv v4) c1)) (co
cexp) (cfv (cv v3) cfa) cdiv)) cmul) (co c1 (csu (co cc0 v2 cfz) (λ v3, co (co (cfv (cv v0) clog) (c
v0) clog) (cv v3) cexp) (cfv (cv v3) cfa) cdiv))
  (@oveq12d (wa (wa (wcel v2 cn0) (wa (wcel (cv v0) crp) (wbr c1 (cv v0) cle))) (wceq (cv v4) c1))
v3) cexp) (cfv (cv v3) cfa) cdiv)) (csu (co cc0 v2 cfz) (λ v3, co (co (cfv (cv v0) clog) (cv v3) cex
   (v218 v0 v4)
   (@sumeq2dv (wa (wa (wcel v2 cn0) (wa (wcel (cv v0) crp) (wbr c1 (cv v0) cle))) (wceq (cv v4) c
fv (cv v3) cfa) cdiv) (λ v3, co (co (cfv (cv v0) clog) (cv v3) cexp) (cfv (cv v3) cfa) cdiv)
    (λ v3, @oveq1d (wa (wa (wa (wcel v2 cn0) (wa (wcel (cv v0) crp) (wbr c1 (cv v0) cle))) (wceq
(cv v3) cexp) (co (cfv (cv v0) clog) (cv v3) cexp) (cfv (cv v3) cfa) cdiv
     (@oveq1d (wa (wa (wa (wcel v2 cn0) (wa (wcel (cv v0) crp) (wbr c1 (cv v0) cle))) (wceq (cv
v0) clog) (cv v3) cexp
      (@adantr (wa (wa (wcel v2 cn0) (wa (wcel (cv v0) crp) (wbr c1 (cv v0) cle))) (wceq (cv v
cc0 v2 cfz))
       (v224 v0 v4))))))
  (@mulid2d (wa (wa (wcel v2 cn0) (wa (wcel (cv v0) crp) (wbr c1 (cv v0) cle))) (wceq (cv v4) c1))
cdiv))
   (@adantr (wa (wcel v2 cn0) (wa (wcel (cv v0) crp) (wbr c1 (cv v0) cle))) (wcel (csu (co cc0 v2
v v4) c1)
    (v184 v0))))))
173 v0)
v4, @a1i (wcel (co (co (cfv (cv v0) clog) v2 cexp) (co (cfv v2 cfa) (csu (co cc0 v2 cfz) (λ v3, co
2 cn0) (wa (wcel (cv v0) crp) (wbr c1 (cv v0) cle)))
(@ovex (co (cfv (cv v0) clog) v2 cexp) (co (cfv v2 cfa) (csu (co cc0 v2 cfz) (λ v3, co (co (cfv (cv

(wa (wcel v2 cn0) (wa (wcel (cv v0) crp) (wbr c1 (cv v0) cle))) (co (co (co (co (csu (co c1 (cfv (cv
0) clog) v2 cexp) (co (cfv v2 cfa) (csu (co cc0 v2 cfz) (λ v3, co (co (cfv (cv v0) clog) (cv v3) cex
```

A harder question

A computer wrote that proof!

It goes on for about 150 kilobytes.

The computer *translated* the proof from another theorem prover called `metamath`, into Lean.

And a human (Mario Carneiro) wrote the `metamath` proof. And the program which did the translation.

An even harder question

So now let's do a thought experiment.

Can we get a computer to prove the Birch and Swinnerton-Dyer conjecture?

Well, `library_search` won't work in Lean.

Because there's no proof in Lean's library.

And ChatGPT won't prove it either.

Because it's not smart enough to put together various ideas on the internet to prove it.

Where are these systems going?

Large language models and interactive theorem provers can both prove mathematical results at school and early undergraduate level.

The problems start when you move beyond that.

The large language models are autonomous, but start talking nonsense.

The theorem provers are super-accurate, but need human guidance.

Note that the weakness of each system is a strength of the other.

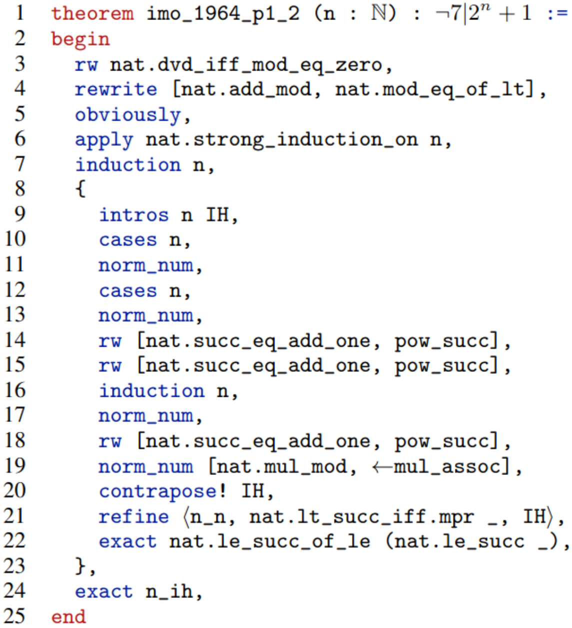
Putting them together?

If each system’s weakness is fixed by the other, then why not put them together?

This can be done!

Let’s see it in action.

Part of an old IMO problem



```
theorem imo_1964_p1_2 (n : ℕ) : ¬ 7 ∣ 2^n + 1 :=
begin
  rw nat.dvd_iff_mod_eq_zero,
  rewrite [nat.add_mod, nat.mod_eq_of_lt],
  obviously,
  apply nat.strong_induction_on n,
  induction n,
  {
    intros n IH,
    cases n,
    norm_num,
    cases n,
    norm_num,
    rw [nat.succ_eq_add_one, pow_succ],
    rw [nat.succ_eq_add_one, pow_succ],
    induction n,
    norm_num,
    rw [nat.succ_eq_add_one, pow_succ],
    norm_num [nat.mul_mod, ←mul_assoc],
    contrapose! IH,
    refine ⟨n_n, nat.lt_succ_iff.mpr _, IH⟩,
    exact nat.le_succ_of_le (nat.le_succ _),
  },
  exact n_ih,
end
```

Large language models and computer code

The previous slide was a large language model writing computer code.

More precisely, it is Meta's LLM solving part of a very easy old IMO problem ($2^n + 1$ is never a multiple of 7).

So we're at. . . school/early undergraduate level. Again.

Where are these systems going?

Of course the *dream* is that one day one of these systems *can* automatically prove the Birch and Swinnerton-Dyer conjecture.

But right now this is more than a dream – this is complete science fiction.

One reason: to *state* the Birch and Swinnerton-Dyer conjecture you need to know that the L-function of the elliptic curve is even *defined* at $s = 1$.

To prove this you need all of the Wiles/Taylor–Wiles machinery and more.

Ideas

I have seen no evidence that current large language models can have important mathematical ideas.

And interactive theorem provers can *understand* modern mathematical ideas, but need to be taught them by a human who knows them already.

Where are these systems going?

From what I’ve said, it seems that we’re stuck at school level mathematics!

But this is not the case at all.

Let me now talk about how computers have been engaging with modern research.

Schemes

I learnt how to use Lean by teaching it what a scheme was, with a bunch of undergraduates as co-authors.

We got a publication out of it!

But the algebraic geometers in my department were not impressed.

I realised we had to aim higher.

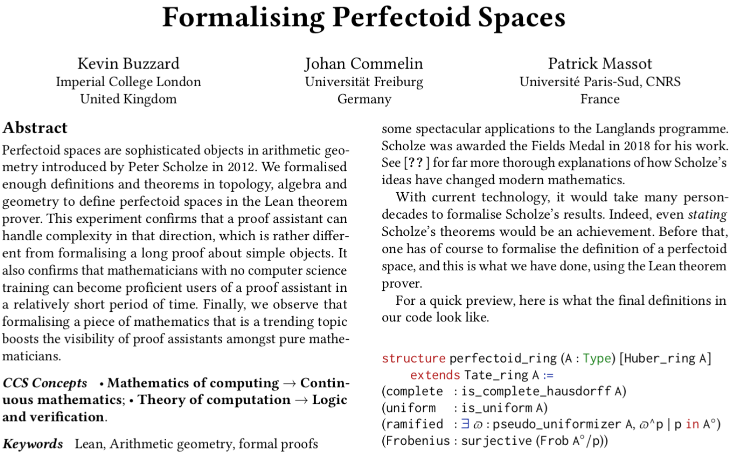
Perfectoid spaces

In 2018 Peter Scholze got a Fields Medal for his discovery/invention of perfectoid spaces and their application in arithmetic geometry.

More recently perfectoid spaces have been used to resolve conjectures in commutative algebra, and in the Langlands Program.

In 2019 Commelin, Massot and I defined perfectoid spaces in the Lean theorem prover.

Lean perfectoid spaces



Formalising Perfectoid Spaces

Kevin Buzzard
Imperial College London
United Kingdom

Johan Commelin
Universität Freiburg
Germany

Patrick Massot
Université Paris-Sud, CNRS
France

Abstract

Perfectoid spaces are sophisticated objects in arithmetic geometry introduced by Peter Scholze in 2012. We formalised enough definitions and theorems in topology, algebra and geometry to define perfectoid spaces in the Lean theorem prover. This experiment confirms that a proof assistant can handle complexity in that direction, which is rather different from formalising a long proof about simple objects. It also confirms that mathematicians with no computer science training can become proficient users of a proof assistant in a relatively short period of time. Finally, we observe that formalising a piece of mathematics that is a trending topic boosts the visibility of proof assistants amongst pure mathematicians.

CCS Concepts • **Mathematics of computing** → **Continuous mathematics**; • **Theory of computation** → **Logic and verification**.

Keywords Lean, Arithmetic geometry, formal proofs

some spectacular applications to the Langlands programme. Scholze was awarded the Fields Medal in 2018 for his work. See [**??**] for far more thorough explanations of how Scholze's ideas have changed modern mathematics.

With current technology, it would take many person-decades to formalise Scholze's results. Indeed, even *stating* Scholze's theorems would be an achievement. Before that, one has of course to formalise the definition of a perfectoid space, and this is what we have done, using the Lean theorem prover.

For a quick preview, here is what the final definitions in our code look like.

```
structure perfectoid_ring (A : Type) [Huber_ring A]
    extends Tate_ring A :=
(complete  : is_complete_hausdorff A)
(uniform   : is_uniform A)
(ramified  : ∃ ϖ : pseudo_uniformizer A, ϖ^p ∣ p in A°)
(Frobenius : surjective (Frob A°/p))
```

Perfectoid spaces

This paper was partly a publicity stunt.

It was an attempt to say “ITPs have arrived.”

It worked great!

The paper has its own web page by the way.

A bunch of young people interested in Scholze’s mathematics showed up on the Lean Zulip chat.

NB the Lean community is *incredibly* grateful to Zulip for hosting our community so well, and enabling it to grow so effectively.

Perfectoid spaces

But there was a catch: we didn’t actually *prove any theorems* about perfectoid spaces.

It was the *theorems* which won Scholze the Fields Medal.

Liquid vector spaces

In 2020, Scholze challenged the ITP community to *prove one of his theorems* – some joint work with Dustin Clausen on condensed mathematics.

You can read about the background of the challenge in a blog post he wrote.

Summary: Scholze was not convinced that the refereeing system was working properly, but wanted to be 100 percent sure of this particular result.

Liquid vector spaces

To prove the theorem in a theorem prover, one would need to first teach the theorem prover about things like

- profinite topological spaces;
- homological algebra (derived functors etc);
- Abelian categories;
- Grothendieck topologies;
- condensed sets;
- Stable homotopy groups of spheres;
- All the prerequisites of the proof.

But (partly because of the perfectoid publicity stunt) we now had the staff.

Lean makes Nature

We were only half way there when our work was featured in Nature!

We finished the project in July 2022.

The “leaders” of the project were Johan Commelin and Adam Topaz.

Commelin managed to completely remove the dependence on the theory of stable homotopy groups of spheres.

The entire body of work was done completely openly, on the Lean Zulip chat, with the code stored in a public repository on GitHub.

Everting spheres

While Commelin and I were doing arithmetic geometry, Massot turned to differential topology.

He led a team which formalised a Lean proof that you could turn a sphere inside-out.

In fact they formalised an instance of Gromov's *h*-principle, and deduced sphere eversion as a corollary.

AI and mathematics

Away from large language models, AI has been making other inroads into research level mathematics.

Let me give some examples.

Knot theory

Researchers from Deep Mind and two knot theorists in Oxford collaborated to prove a new theorem in knot theory.

Paper:
The signature and cusp geometry of hyperbolic knots, by Davies, Juhász, Lackenby and Tomasev (2021).

Tables of knots and their invariants (discrete and continuous) were fed to a neural network.

The network was asked to try and discover relations between the invariants.

Knot theory

Human experimentation led to the following question: can one predict the *signature* $\sigma(K)$ of a knot (an integer, and an algebraic invariant) from standard hyperbolic invariants of the knot (real or complex numbers)?

A neural network was given a database containing the signatures and geometric invariants of millions of knots.

The neural network was able to “learn” how to predict the signature of a knot, given the geometric invariants, with high accuracy.

Analysis of the network tells you which geometric invariants are “the most important” in the algorithm.

Further experimentation gave evidence for a linear correlation between a new geometric invariant (the “slope”) of a knot and the signature.

Knot theory

Humans then took over and the idea became a conjecture relating the slope of a knot and its signature.

And then human knot theorists proved it.

Theorem

There exists a constant c such that for any hyperbolic knot K,

$$|2\sigma(K) - \mathrm{slope}(K)| \leq c\,\mathrm{vol}(K)\,\mathrm{inj}(K)^{-3}.$$

The theorem is surprising because it relates algebraic invariants to geometric invariants.

Summary: a neural network told humans *where to look* in their data for new theorems.

The combinatorial invariance conjecture

Take two elements of a symmetric group.

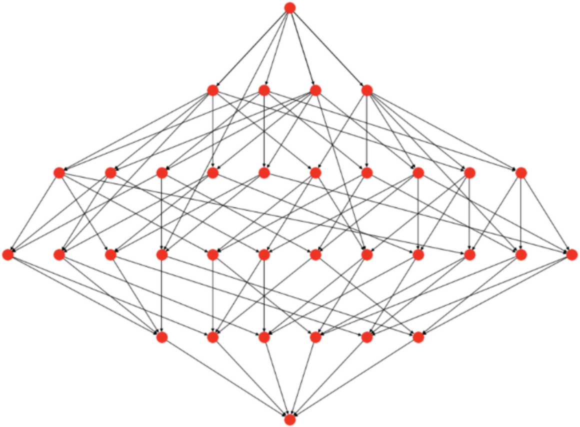
Invariants associated to this data:

1) the Bruhat interval graph (unwieldy object, easy to compute).

2) The Kazhdan–Lusztig polynomial (simple object, hard to compute).

The conjecture is that the KL polynomial can be computed from the Bruhat interval graph.

The combinatorial invariance conjecture



$\mapsto 1+q^2$

The combinatorial invariance conjecture

Again one has access to a large dataset.

The idea: pre-process the interval graph, and then use a neural network to try and predict the KL polynomial from the processed data.

Sometimes the neural network would get very good at predicting the polynomial.

Humans learnt from the network that edges in the graph corresponding to extremal reflections seemed to be important.

Again, a theorem was isolated and proved (by Geordie Williamson).

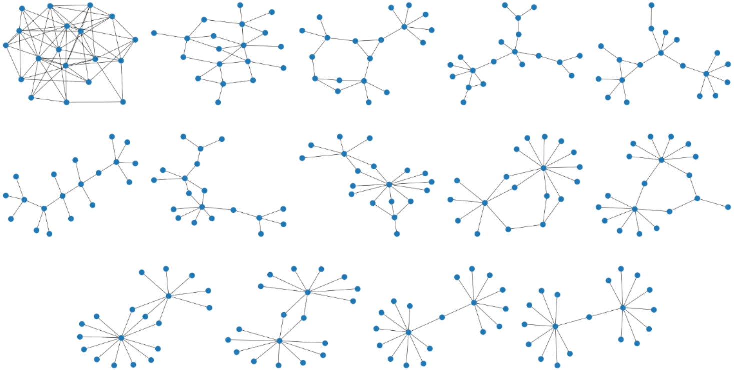
Applications of neural networks to mathematics

Final example: “Constructions in combinatorics via neural networks”, by Adam Zsolt Wagner.

Conjecture from 2010: If G is a connected graph with $n \geq 3$ vertices, with largest eigenvalue λ_1 and matching number μ, then

$$\lambda_1 + \mu \geq 1 + \sqrt{n-1}.$$

Techique: fix n, and tell the neural network to try and minimise $\lambda_1 + \mu$ by switching edges on and off in a random graph.



Applications of neural networks to mathematics

The neural net discovers structure which makes $\lambda_1 + \mu$ small.

The graph below is a counterexample to the conjecture:

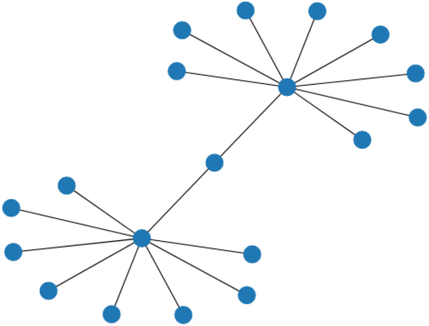


Figure 4: A graph on 19 vertices satisfying $\lambda_1 + \mu < \sqrt{n-1} + 1$.

Contributing as a mathematician

I'll end by briefly explaining what my PhD students and post-docs are doing nowadays.

We're not directly involved in the AI side of things.

We want to teach the system things like the definitions of the important objects in number theory.

My students to go other courses and see what is happening in their areas of interest.

They then teach parts of the courses to Lean.

Examples

Recent examples:

- Ashvni Narayanan taught Lean about (abelian) p-adic L-functions. She proved that their values at negative integers were related to Bernoulli numbers.
- David Angdinata is well on the way to proving the Mordell–Weil theorem for elliptic curves.
- Amelia Livingston has defined group and Galois cohomology, and proved theorems like Hilbert 90. Etale cohomology is next.
- Oliver Nash has defined Lie algebras and is working on the classification of semisimple Lie algebras via root systems with Deepro Choudhury (an undergraduate).

Examples

- María Inés de Frutos Fernández has defined the adeles of a global field, related them to class groups, and stated the main theorem of global class field theory.
- María Inés has also defined Fontaine's period ring B_{HT} and has nearly finished B_{dR} and B_{cris} (with Antoine Chambert-Loir).
- Jujian Zhang defined projective schemes and is now working on flat modules.

Research

Are these people "doing research"?

They are not proving new theorems.

They are discovering the *right way* to think about theorems.

They are clarifying and simplifying the literature (example: Nash's "Engelian Lie algebras").

More importantly, they are building a gigantic *digitised* database of modern mathematics.

It is in a format which *computers can understand*.

I am convinced that people will find all sorts of uses for this.

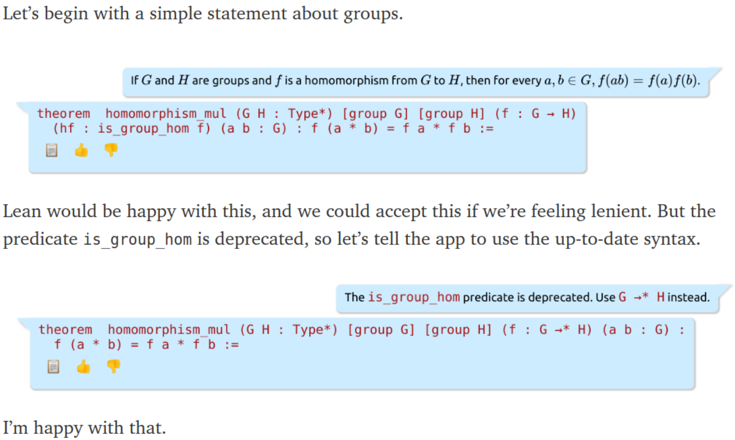
Translating pdf into Lean

Simultaneously, AI experts are working on translating human language into Lean.

Translating proofs into Lean is *really hard* right now, and may remain really hard.

But they're getting very good at translating *statements*.

Lean chat



Let’s begin with a simple statement about groups.

If G and H are groups and f is a homomorphism from G to H, then for every $a, b \in G$, $f(ab) = f(a)f(b)$.

```
theorem homomorphism_mul (G H : Type*) [group G] [group H] (f : G → H)
  (hf : is_group_hom f) (a b : G) : f (a * b) = f a * f b :=
```

Lean would be happy with this, and we could accept this if we’re feeling lenient. But the predicate `is_group_hom` is deprecated, so let’s tell the app to use the up-to-date syntax.

The `is_group_hom` predicate is deprecated. Use `G →* H` instead.

```
theorem homomorphism_mul (G H : Type*) [group G] [group H] (f : G →* H) (a b : G) :
  f (a * b) = f a * f b :=
```

I’m happy with that.

(see more at the blog post by Zhangir Azerbayev)

Possible future goals

Can we train an AI on the Stacks Project (human-written algebraic geometry database) and translate the *statements* of the theorems into Lean? Can we make an *accurate* algebraic geometry chatbot? Or any other topic you choose?

Can we build an interactive textbook at graduate level? See Patrick Massot's amazing demo at a conference at UCLA last month (live web page here until Massot takes it down)

Can we make a front end for Lean which is actually *easy to use* for mathematicians?

Can we prove Fermat's Last Theorem in Lean?

Thanks a lot for listening!

hashtag ad: if you're a mathematics undergraduate and interested in experimenting with Lean, you could try looking at my undergraduate course notes, and if you have any questions then ping me and ask for an invite to the Xena Project Discord where you can ask basic questions to other maths undergrads and PhD students.