Mathematics and the computer

Kevin Buzzard, Imperial College London

EECS 2023, 3rd April 2023

Before we start

Thank you very much to Siddhartha Gadgil and IISc for the invitation to India, and to Rajesh Sundaresan for the invitation to speak here.

Overview of the talk

Who am I? I'm a pure mathematician (an algebraic number theorist), working at Imperial College in London England.

But today I'm going to talk about whether machines will one day be able to do my job.

What exactly *is* my job?

Pure mathematicians

One aspect of my job is *research*.

What does research look like in pure mathematics?

Pure mathematics research

There are many hard unsolved problems in pure mathematics.

Examples: the Riemann Hypothesis, the $3n + 1$ problem, the Goldbach conjecture, the Birch and Swinnerton-Dyer conjecture, The P=NP problem, the Langlands conjectures, . . .

Some are easy to explain.

Goldbach conjecture: every even number which is 4 or more, is the sum of two prime numbers.

It's easy to state, but seems hard to prove.

Pure mathematics research

Here's another example of a difficult problem.

It's easy to check that $3^2 + 4^2 = 5^2$, $5^2 + 12^2 = 13^2$, $8^2 + 15^2 = 17^2$,...

A conjecture attributed to Fermat ("Fermat's Last Theorem") says that there are no positive integers x, y, z and $n \geq 3$ such that $x^n + y^n = z^n$.

The conjecture was open for 351 years.

It was proved in 1994 by Wiles and Taylor/Wiles.

Fermat's Last Theorem

Theorem (Wiles + Taylor/Wiles, 1994): No positive integer solutions to $x^n + y^n = z^n$ if $n \geq 3$.

The statement uses positive integers, addition and exponentiation.

The proof uses elliptic curves, modular forms, Galois representations, deformation theory, finite flat group schemes, algebraic geometry, harmonic analysis, abelian varieties, ...

Essentially no mathematical "machinery" needed to *state* the theorem.

A lot of machinery needed in all known proofs.

Pure mathematics research

Some problems are hard to even state.

All of the machinery which went into the proof of Fermat's Last Theorem is needed before one can even rigorously *state* the Birch and Swinnerton-Dyer conjecture.

Pure mathematics research

A researcher might work on solving a hard problem.

Or they might just work on moving closer to solving a hard problem.

So where do computers come in?

Most hard problems in pure mathematics *cannot be solved by a computation*.

Pure mathematics research

Example: the Goldbach conjecture.

Conjecture: Every even number $n \geq 4$ is the sum of two prime numbers.

Examples: $4 = 2 + 2$, $6 = 3 + 3$, $8 = 3 + 5$, $10 = 3 + 7 = 5 + 5$, $12 = 5 + 7, \ldots$

The statement has been checked by computers for all even numbers n in the range $4 \leq n \leq 4 \times 10^{18}$.

So we are zero percent of the way there.

Computers in mathematics

Again: the problem with traditional uses of computers in pure mathematics:

Most hard problems *cannot* be solved with a finite computation.

In fact many *easy* problems in mathematics cannot be solved with a finite computation!

Let's consider an example going back over 2000 years.

A theorem of Euclid: There are infinitely many prime numbers.

Infinitely many prime numbers

It’s easy to write Python code which will print out prime number after prime number.

That is *evidence* that there are infinitely many prime numbers, but it is not a *proof*.

But nowadays we *can* use computers to prove that there are infinitely many prime numbers!

Let me now talk about two ways to do this.

Large language models

We can ask a large language model such as ChatGPT to prove that there are infinitely many prime numbers!

SP

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.

ChatGPT

ChatGPT gave me a “7 out of 10” proof (the logic in the composite case is confused).

But I’m sure sometimes it will give you a “10 out of 10” proof.

It’s a bit like asking a class full of first year undergraduates to prove it.

What about a harder question?

Let's ask ChatGPT a harder mathematical problem.

Let's ask it to prove that there are infinitely many primes which end in 7 (for example $7, 17, 37, 67, 97, \ldots$).

It's true, but *much* harder to prove.

It was only solved just over 100 years ago.

Primes ending in 7

SP

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.

Chat-GPT

This is a disaster.

The system does what an undergraduate would try.

It knows a method, and it tries to generalise it.

The method *does* generalize to, for example, proving there are infinitely many primes which are 3 mod 4.

But it doesn't work here: a large composite number ending in 7 can be the product of two smaller primes which don't.

The system doesn't notice.

Like an undergraduate, it just "tries to get the marks".

One last example

I asked ChatGPT to prove that there were infinitely many even numbers ending in 7.

SP: Can you prove that there are infinitely many even numbers which end in 7?

Yes, it can be proven that there are infinitely many even numbers which end in 7.

To see why, consider that any even number can be written in the form 2n, where n is an integer. To find even numbers that end in 7, we can write this as 10n + 7, since adding 7 to a number that ends in 0 gives a number that ends in 7.

Now suppose there were only finitely many even numbers that end in 7, say k of them. We can denote these numbers as 10n1 + 7, 10n2 + 7, ..., 10nk + 7.

Let M be the product of all of these numbers, so M = (10n1 + 7)(10n2 + 7)...(10nk + 7). Then we can write M as:

M = 10k(n1 + n2 + ... + nk) + 7(...)

where the (...) represents the product of all the other terms that do not end in 7.

Note that the first term, 10k(n1 + n2 + ... + nk), is even because it is a multiple of 10. Since M is the product of an even number of even numbers, it is even.

However, the last digit of M is 7, since every term in the product ends in 7. This is a contradiction, since even numbers cannot end in 7. Therefore, our assumption that there were only finitely many even numbers that end in 7 must be false.

Thus, there must be infinitely many even numbers that end in 7.

ChatGPT

So is ChatGPT going to take my job?

Right now, I can't see it scaling to serious mathematics.

These systems have become *exponentially better* recently.

They have got really good at *lots of things which aren't mathematics*.

Some computer scientists think that this exponential increase will continue.

Some of them say these systems will be proving new theorems by 2030.

Maybe they're right, but I don't see it right now.

They're already training on the internet! How can they scale up?

Interactive theorem provers

Now let's use a computer in a different way, to again prove that there are infinitely many prime numbers.

This is Lean, an interactive theorem prover developed by Leo de Moura at Microsoft Research.

[Lean demo]

Interactive theorem provers

Lean proved it “automatically”!

Or did it?

Lean’s mathematics library `mathlib` is over a million lines of code, containing 110,000 theorems and counting.

All those theorems were typed in manually by humans.

Let’s take a look at the proof which Lean found.

[examine Lean proof]

Interactive theorem provers

Lean can also prove that there are infinitely many primes which end in 7!

The proof is not in its mathematics library though.

It looks like this:

Infinitely many primes end in 7.

```
(@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
```

Infinitely many primes end in 7.

That proof was written by a computer!

It is tens of thousands of lines of incomprehensible (to me) computer code.

The computer was *translating* a proof from the maths library of `metamath`, another interactive theorem prover.

And the `metamath` proof was written by a human (Mario Carneiro).

Carneiro also wrote the program which translated the proof into Lean.

Ultimately it's still humans writing the proof.

How far can we go with Lean?

How far can interactive theorem provers go?

We’ve seen them get to the year 1900.

Can get they get to 2020?

Yes they can!

Artificial Intelligence and Mathematics

My experiments in automating mathematics.

- Home
- Automating Mathematics group
- ProvingGround
- Siddhartha Gadgil's homepage
- Superficial

Formalizing Gardam's disproof of Kaplansky's Unit Conjecture

Tue, Jun 7, 2022

A little over an year ago, Giles Gardam disproved a long-standing conjecture, often called the *Kaplansky Unit Conjecture*. This was a striking result – the statement was a simple and basic one, it had a long history, and it was one of a cluster of related conjectures with important relations to many areas (including the Whitehead conjecture in topology). Gardam's work is published in the *Annals of Mathematics*.

Anand Rao Tadipatri and I have formalized Gardam's disproof in **lean 4**. We used **lean 4** as a proof assistant, but also took advantage of its being a (full fledged and really nice) programming language. This post is an account of the Unit conjecture and our formalization of it.

Siddhartha Gadgil and Anand Rao Tadipatri are both in the audience!

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))
```

nature

Explore content About the journal Publish with us Subscribe

nature > news > article

NEWS | 18 June 2021

Mathematicians welcome computer-assisted proof in 'grand unification' theory

Proof-assistant software handles an abstract concept at the cutting edge of research, revealing a bigger role for software in mathematics.

Davide Castelvecchi

LLMs and ITPs

So what did we see so far in this talk?

A large language model can write mathematics by itself.

But beyond some point around early undergraduate level, the output is typically nonsense.

Interactive theorem provers don't make mistakes*.

They can be led to the boundary of modern mathematics.

But the code must be written by humans.

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

So let's put them together!

The result

```
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
```

IMO 1964

That is a large language model trained by Meta to write Lean code.

It has *automatically* proved half of a very old IMO problem.

It is the first time in this talk we have seen mathematics truly generated by a machine.

It is at high school level :-(

This is state of the art :-(

Conclusions so far

I am not scared of being replaced by a machine right now.

I already have a system which can do mathematics at the level of an undergraduate.

It's the 300 undergraduates which I teach every year.

Right now we need to ask an easier question.

What about teaching?

Another part of my job is teaching.

What does this look like?

I give undergraduate lectures.

And I have one to one conversations with students who have questions.

During Covid, all my lectures got recorded.

So my university already has them.

Undergraduate teaching

So what is left is the one on one conversations.

Will a large language model be able to replace me?

Maybe no, maybe yes.

We need to wait and see.

But it's certainly not out of the question.

It will be easier for an "introduction to proofs" course (1st year undergrad) than for a real manifolds course (final year undergrad / 1st year MSc).

What about teaching?

A more challenging question: what about supervising PhD students?

What does this look like in, say, number theory?

For a starting PhD student, you might ask them to begin with something incremental.

For example “paper *X* proves a theorem about `foo`s, but the techniques should work for generalized `foo`s, so can you generalize paper *X* for me?”

This can be a good starting point for a PhD student in pure mathematics.

What this boils down to

Ultimately the student is faced with the following kind of question.

The `foo` theory paper claims (perhaps without proof) that a certain claim is true for `foo`s.

The student needs to figure out whether the analogous generalized claim is true for generalized `foo`s.

This is *exactly* the problem which Chat-GPT was so lousy at.

It knew a trick for primes, so it just *stated* that some kind of generalization worked for primes ending in 7, because it was naively pattern-matching without understanding.

Can we do better?

So how does a first year PhD student find out the answers to questions such as “does this claim, which is true for `foo`s, also hold for generalised `foo`s”?

They might

- ask more experienced PhD students;
- ask a post-doc;
- ask their PhD advisor;
- ask google;
- ask `mathoverflow`;
- try to work it out themselves.

Can we do better?

Note that the answer may well be “standard” (e.g. in a standard hard textbook or paper, or in your PhD advisor’s head because it’s “well-known”).

What do we want?

One thing I believe: if large language models can't solve this kind of task, then they *surely* can't prove interesting new theorems.

This is why I think it's an interesting benchmark.

What do we want?

I think that interactive theorem provers could be used to help make such a system.

Back end: a big database of formalized theorems and counterexamples.

Counterexample finders and “hammers” (tools which put theorems together) already exist.

Human-friendly front end.

Can we have this?

Counterexample finders and hammers: sure.

Human-friendly front end: sure.

Gigantic database of modern results in pure mathematics in a form understandable by an interactive theorem prover: Currently a problem.

Why? Computers cannot read the human literature.

And right now humans are not rewarded for doing this kind of work manually :-(

Social engineering

We have been proposing *big challenges* in the Lean mathematics community.

My observation: if you engage with fashionable mathematics, then it's not hard to find people who will help you teach it to the computer.

The community solves the challenges, and our database gets bigger.

But it is still *highly incomplete.*

Machines translating the literature

Here is an example of a database: the Stacks Project.

Thousands of theorems in algebraic geometry, all on the internet, written in LaTeX.

Problem: right now language models struggle to read human proofs and translate them into Lean.

But they're getting very good at translating *theorem 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). Gadgil (IISc) is also thinking about this.

Summary

Mathematicians should be aware that new tools are coming.

I don't see evidence, yet, that large language models *alone* are the solution for mathematics beyond elementary undergraduate level.

Writing code in them which corresponds to statements and proofs of hard theorems is fun, and also might be important.

One way into the subject: choose a mathematical statement or proof you want to formalise, and ask an expert how to do it or where to get help.

One last thing.

Interactive theorem provers can be used to power error-free textbooks!

See here (until Patrick Massot takes it down) (joint work Patrick Massot / Kyle Miller)

Thanks for coming!

I've been Kevin Buzzard; I'll be in the maths department all week, giving more mathematical talks and Lean demos for mathematicians, and talking to people.

Introductory talks Tuesday 9:30-11 and and 2-3 in Maths department lecture hall 3.

Wednesday 9-10 I'll do some undergraduate level number theory also in lecture hall 3.

Come find me if you have questions! (Siddhartha Gadgil will know where I am)