

Machine assisted proofs

Terence Tao

University of California, Los Angeles

Jan 3, 2024

Mathematicians have used **machines** and **computers** to assist them in their research for centuries.

One of the earliest uses of (human) computers in mathematics was to build tables, such as **logarithm tables** and **trigonometric tables**.

	Sinus	Tangens	Secans
14			
31	2506616	2589280	10329781
32	2509432	2592784	10330559
33	2512248	2595488	10331339
34	2515063	2598292	10332119
35	2517879	2601099	10332901
36	2520694	2604005	10333683
37	2523508	2607011	10334467
38	2526323	2611018	10335251
39	2529137	2614126	10336037
40	2531952	2617234	10336823
41	2534766	2620342	10337611
42	2537579	2623451	10338399
43	2540393	2626560	10339188
44	2543206	2629670	10339979
45	2546019	2632780	10340770
46	2548832	2635891	10341563
47	2551645	2639002	10342356
48	2554458	2642114	10343151
49	2557270	2645226	10343945
50	2560082	2648339	10344741
51	2562894	2651452	10345540
52	2565705	2654566	10346338
53	2568517	2657680	10347138
54	2571328	2660794	10347938
55	2574139	2663909	10348740
56	2576950	2667025	10349542
57	2579760	2670141	10350346
58	2582570	2673257	10351150
59	2585381	2676374	10351955
60	2588190	2679492	10352762

Modern tables are still important today, particularly in **experimental mathematics**. Some prominent examples:

- In the late 18th century, Legendre and Gauss used tables of prime numbers to conjecture the **prime number theorem**.
- In the 1960s, Birch and Swinnerton-Dyer used tables of elliptic curve data to formulate the **Birch and Swinnerton-Dyer conjecture**.
- The **Online Encyclopedia of Integer Sequences (OEIS)** has helped discover (or rediscover) countless mathematical relationships.
- Large databases of mathematical objects can be used as input for machine learning algorithms (more on this later).

Another venerable mathematical use of machines is in **scientific computing**. For instance, in the 1920s, Hendrik Lorentz assembled a team of human computers to model the fluid flow around the *Afsluitdijk* - a major dam then under construction in the Netherlands. (An early use of **floating point arithmetic**!)

t	- 10000	- 5000	0	5000	10000	15000	20000	25000	sec
$h_{II} \dots \dots \dots$	31,7	45,1	35,7	10,8	1,8	- 6,5	- 17,0	- 5,1	$\times 10^{-12}$
$s_{III} = \dot{h}_{II} \dots \dots \dots$	80	4	- 35	- 34	- 17	- 19	1	20	$\times 10^{-10}$
$\bar{s}_{II} \dots \dots \dots$	- 4	- 12	- 9	- 7	- 12	- 8	3	5	$\times 10^{-10}$
$\pm 2 \bar{s}_1^2 \dots \dots \dots$	40	60	48	6	1	0	15	50	$\times 10^{-10}$
$\pm 2 \bar{s}_0 \bar{s}_{II} \dots \dots \dots$	92	73	15	1	- 40	- 81	- 59	- 5	$\times 10^{-10}$
$h_{III} \dots \dots \dots$	128	121	49	0	- 51	- 89	- 41	50	$\times 10^{-10}$
$\bar{h}_0 \dots \dots \dots$	5,19	7,49	10,67	12,11	12,45	12,62	11,95	9,92	$\times 10^{-4}$
$-\bar{l} \bar{h}_I \dots \dots \dots$	- 0,24	- 1,38	- 1,74	- 1,69	- 1,76	- 0,85	0,30	1,30	$\times 10^{-4}$
$\frac{\bar{l}^2}{2} \bar{h}_{II} \dots \dots \dots$	0,45	0,64	0,51	0,15	0,08	- 0,09	- 0,24	- 0,07	$\times 10^{-4}$
$-\frac{\bar{l}^3}{6} \bar{h}_{III} \dots \dots \dots$	- 0,10	- 0,10	- 0,04	0,00	0,04	0,07	0,03	- 0,04	$\times 10^{-4}$
$\bar{h}_I \dots \dots \dots$	5,30	6,65	9,40	10,57	10,76	11,75	12,04	11,11	$\times 10^{-4}$
$h_s \dots \dots \dots$	98	123	173	194	199	217	222	206	em
$\bar{s}_0 \dots \dots \dots$	3,30	4,11	4,25	4,04	4,01	3,32	2,32	0,26	$\times 10^{-4}$
$-\bar{l} \bar{s}_I \dots \dots \dots$	- 0,76	- 0,92	- 0,78	- 0,30	- 0,09	0,08	0,46	0,84	$\times 10^{-4}$
$\frac{\bar{l}^2}{2} \bar{s}_{II} \dots \dots \dots$	0,20	0,18	0,03	0,00	- 0,07	- 0,17	- 0,18	- 0,13	$\times 10^{-4}$
$-\frac{\bar{l}^3}{6} \bar{s}_{III} \dots \dots \dots$	- 0,02	- 0,00	0,03	0,03	0,01	0,02	- 0,00	- 0,02	$\times 10^{-4}$
$\bar{s}_I \dots \dots \dots$	2,72	3,32	3,53	3,77	3,86	3,25	2,60	0,95	$\times 10^{-4}$
$S_s \dots \dots \dots$	67	81	87	92	94	80	64	23	$\times 10^9$

- Modern **computer algebra systems** (e.g., Magma, SAGEMath, Mathematica, Maple, etc.), as well as more general-purpose programming languages, are now routinely used to perform symbolic computations in algebra, analysis, geometry, number theory, and many other branches of mathematics.
- Using tools such as **interval arithmetic** in place of floating point arithmetic, one can (sometimes) avoid the possibility of roundoff errors when using these tools, possibly at the expense of increased runtime.
- A relative of computer algebra systems are **satisfiability (SAT) solvers** and **satisfiability modulo theories (SMT) solvers**, which can perform complex logical deductions of conclusions from certain restricted sets of hypotheses, and generate proof certificates for each such deduction.
- Of course, satisfiability is a **NP-complete** problem, so these solvers do not scale past a certain point.

An example of a result proved using a SAT solver:

Boolean Pythagorean triples theorem (Heule–Kullmann–Marek, 2016)

The set $\{1, \dots, 7824\}$ can be partitioned into two classes, neither of which contains a Pythagorean triple (a, b, c) with $a^2 + b^2 = c^2$; however, this is not possible for $\{1, \dots, 7825\}$.

The proof required 4 CPU-years of computation and generated a 200 terabyte propositional proof, which was later compressed to 68 gigabytes.

But there are also promising new ways to use computers in mathematical research (besides routine tasks such as writing papers, email, web search, etc.):

- **Machine learning algorithms** can be used to discover new mathematical relationships, or generate potential examples or counterexamples for mathematical problems.
- **Large language models** can (potentially) be used to make other tools easier and faster to use; they can also suggest proof strategies or related work, and even generate (simple) proofs directly.
- **Formal proof assistants** can be used to verify proofs (as well as the output of large language models), allow truly large-scale mathematical collaborations, and help build data sets to train the aforementioned machine learning algorithms.

There appears to be a great potential for these tools to complement each other, with one tool counteracting the weaknesses of another.

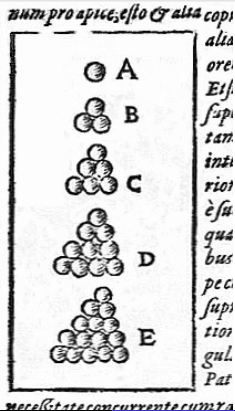
Proof assistants

- Early machine-assisted proofs, such as Appel and Haken's proof of the **four color theorem** in 1976, did not use a fully developed formal proof assistant; instead, the proof was verified by a combination of computer and human checking, and was still prone to various (fixable) errors.
- A truly computer-verifiable proof of the four color theorem was first produced in 1996 by Robertson, Sanders, Seymour, and Thomas.
- A fully formalized proof of the theorem was given in 2005 by Werner and Gonthier, in the proof assistant **Coq**.

Another celebrated use of machine assistance was in the proof of the **Kepler conjecture**:

Kepler Conjecture (Kepler, 1611)

The densest packing in \mathbf{R}^3 by unit balls is that given by the hexagonal close packing (or the cubic close packing), both of which have a density of $\frac{\pi}{3\sqrt{2}} = 0.7404\dots$



- *A priori*, this is an optimization problem in infinitely many variables (the locations of the centers of the balls), which is not obviously amenable to computer verification.
- Each ball in a packing comes with a **Voronoi cell** - the polyhedron of all points closer to the center of that ball than any other center.
- In 1951, Toth observed that certain weighted inequalities on volumes of finitely many Voronoi cells would lead to upper bounds on the density of a sphere packing.
- In principle, this could lead to a computer-verified proof, if one could locate such a weighted inequality that gave the sharp upper bound of $\frac{\pi}{3\sqrt{2}}$.
- However, despite many high-profile attempts, no convincing proof along these lines could be located.

- Over the course of many papers from 1994 to 1998, Thomas Hales proposed a complicated modification of Toth's strategy involving more sophisticated polyhedral partitions of space than the Voronoi decomposition, and an assignment of a "score" function to each polyhedron.
- In principle, linear programming could be used to convert inequalities between scores of clusters of polyhedra into upper bounds on the density of sphere packing.

The method was flexible - perhaps *too* flexible.

“Samuel Ferguson and I realized that every time we encountered difficulties in solving the minimization problem, we could adjust the scoring function σ to skirt the difficulty. The function σ became more complicated, but with each change we cut months – or even years – from our work. This incessant fiddling was unpopular with my colleagues. Every time I presented my work in progress at a conference, I was minimizing a different function. Even worse, the function was mildly incompatible with what I did in earlier papers, and this required going back and patching the earlier papers.” (Hales, 2006)

- By 1998, Hales and Ferguson finally announced that the project was successful, deriving the Kepler conjecture from a linear programming computation applied to a carefully chosen optimization problem in 150 variables.
- While initially the project was not conceived as a computer-assisted one, machine calculations became increasingly necessary as the project developed. At the time of the announcement, the proof consisted of 250 pages of notes and 3 gigabytes of computer programs, data and results.
- The refereeing process at *Annals of Mathematics* took four years and a panel of twelve referees. The panel were “99% certain” of the correctness of the proof, but could not certify the correctness of the computer calculations.
- In the end, the paper was accepted for *Annals* (initially with a caveat, that has since been removed).

- In order to dispel doubts about the proof, in 2003 Hales initiated the **Flyspeck project** to prove the Kepler conjecture in the language of a proof assistant, so that the verification could be performed automatically by a computer.
- The project was initially estimated to take twenty years, but after the combined efforts Hales and 21 collaborators, the project was completed in 2014, following a formal blueprint produced by Hales in 2012, and using a combination of the proof assistants **Isabelle** and **HOL Light**.



Forum of Mathematics, Pi (2017), Vol. 5, e2, 29 pages
doi:10.1017/fmp.2017.1



1

A FORMAL PROOF OF THE KEPLER CONJECTURE

THOMAS HALES¹, MARK ADAMS^{2,3}, GERTRUD BAUER⁴,
TAT DAT DANG⁵, JOHN HARRISON⁶, LE TRUONG HOANG⁷,
CEZARY KALISZYK⁸, VICTOR MAGRON⁹, SEAN MCLAUGHLIN¹⁰,
TAT THANG NGUYEN⁷, QUANG TRUONG NGUYEN¹,
TOBIAS NIPKOW¹¹, STEVEN OBUA¹², JOSEPH PLESO¹³, JASON RUTE¹⁴,
ALEXEY SOLOVYEV¹⁵, THI HOAI AN TA⁷, NAM TRUNG TRAN⁷,
THI DIEP TRIEU¹⁶, JOSEF URBAN¹⁷, KY VU¹⁸ and
ROLAND ZUMKELLER¹⁹

Another recent formal proof verification project is the **liquid tensor experiment** of Peter Scholze.

- This experiment involves Scholze's theory of **condensed mathematics**, which aims to repair some deficiencies with categories of topological spaces (e.g., topological abelian groups) by replacing them with a condensed analogue (such as the category of condensed abelian groups).
- A fundamental result in the subject is a certain vanishing theorem

$$\mathrm{Ext}_{\mathrm{Cond}(\mathrm{Ab})}^i(\mathcal{M}_{p'}(S), V) = 0$$

of Clausen and Scholze in 2019, for all p -Banach spaces V and profinite sets S , in the category of condensed abelian groups.

- This theorem is essential to any attempt to study functional analysis through the algebraic lens of condensed mathematics.

Some quotes from Scholze in a blog post from December 2020:

- “ I spent much of 2019 obsessed with the proof of this theorem, almost getting crazy over it. In the end, we were able to get an argument pinned down on paper, but I think nobody else has dared to look at the details of this, and so I still have some small lingering doubts.”
- “with this theorem, the hope that the condensed formalism can be fruitfully applied to real functional analysis stands or falls. I think the theorem is of utmost foundational importance, so being 99.9% sure is not enough.”
- “while I was very happy to see many study groups on condensed mathematics throughout the world, to my knowledge all of them have stopped short of this proof. (Yes, this proof is not much fun...)”
- “I think this may be my most important theorem to date. ... Better be sure it's correct...”

In this blog post, Scholze proposed the “liquid tensor experiment” to formalize this result in the proof assistant Lean.

- This required adding several foundational mathematical theories (e.g., homological algebra, sheaf and topos theory, etc.) to Lean’s `mathlib` library.
- A key subtheorem was formalized in May of 2021, and the full theorem was formalized in July of 2022, by a large group led by Johan Commelin and Adam Topaz, and with constant help from Scholze.

- In addition to formally verifying the proof, numerous minor errors were found and fixed, and multiple small simplifications discovered.
- One major simplification encountered during the process was to replace the difficult theory of the **Breen–Deligne resolution** with a simpler construction that was weaker, but easier to verify the properties of, and still useful for this project (and likely for other problems also).
- The substantial expansion of Lean's `mathlib` library has allowed for other formalization projects to become much easier.

The formalization proceeded in parallel with creating a human-readable “blueprint” of the proof that synthesizes the formal and informal proof in an interactive fashion, and has allowed both Scholze and others to understand the proof much better.



Blueprint for the Liquid Tensor Experiment

Introduction

1 First part ▶

2 Second part ▼

2.1 Variants of normed groups

2.2 Spaces of Measures

2.3 The MacLane Q' -construction

2.4 Condensed abelian groups

3 Bibliography

Section 1 graph

Section 2 graph

In particular, for torsion-free abelian groups A , there is a functorial isomorphism

$$H_i(Q'(A)) \cong H_i(Q'(\mathbb{Z})) \otimes A.$$

As the proof shows, we do not really need the Q' -construction here: Any Breen–Deligne package will do.

Proof ▶

We note that by functoriality of the Q' -construction, it can also be applied to condensed abelian groups.

Corollary 2.3.2 ✓

For torsion-free condensed abelian groups A , there is a natural isomorphism

$$H_i(Q'(A)) \cong H_i(Q'(\mathbb{Z})) \otimes A$$

of condensed abelian groups.

Here, we only need to be able to tensor condensed abelian groups with (abstract) abelian groups. (With more effort, one could prove that $H_i(Q'(\mathbb{Z}))$ is even finitely generated.) In that case, the tensor product functor can be defined very naively by tensoring the values at any S with the given abstract abelian group.

Proof ▼

Evaluating at $S \in \text{ExtrDisc}$, we note that $S \mapsto H_i(Q'(A(S)))$ is already a condensed abelian group, and agrees with $H_i(Q'(\mathbb{Z})) \otimes A(S)$. Thus, the same is true after sheafification.

Partly inspired by this work, software tools are now being developed to automatically convert formal proofs (in languages such as Lean) into human-readable interactive proofs.

Theorem (continuous_of_dense). Let X be a topological space and let Y be a regular topological space. Let A be a dense subset of X . Let $f : X \rightarrow Y$ be a function. Assume that for all elements x of X , f is continuous at x within A . Then f is continuous.

Proof. \circ Let x be an element of X . \circ One can see it suffices to prove that for all closed neighborhoods V' of $f(x)$, there exists a neighborhood U of x such that $f[U] \subseteq V'$. \circ We see this by the following argument:

\circ Let V' be a neighborhood of $f(x)$. \circ Using our assumption that Y is regular and our assumption that V' is a neighborhood of $f(x)$, lemma `RegularSpace.closed_nhd_basis` applied to $f(x)$ and V' proves one can obtain a closed neighborhood W of $f(x)$ such that $W \subseteq V'$. \circ One can obtain a neighborhood U of x such that $f[U] \subseteq W$. \circ We will show that U is suitable by proving that U is a neighborhood of x and $f[U] \subseteq V'$. By assumption, U is a neighborhood of x . \circ Using our assumption that $f[U] \subseteq W$ and our assumption that $W \subseteq V'$ proves $f[U] \subseteq V'$. \circ Let V' be a closed neighborhood of $f(x)$. \circ We obtain an open neighborhood V of x such that $f[V \cap A] \subseteq V'$. \circ We will show that V is suitable by proving that V is a neighborhood of x and $f[V] \subseteq V'$. By assumption, V is a neighborhood of x . \circ Let z be an element of V . \circ One can see that $f(z_1)$ tends to $f(z)$ as z_1 tends to z within $V \cap A$. \bullet

Consider the following:

$$\begin{aligned} f(z) &\in \overline{f[V \cap A]} \\ &\subseteq \overline{V'} \\ &= V' \end{aligned}$$

- \circ Using our assumption that $f(z_1)$ tends to $f(z)$ as z_1 tends to z within $V \cap A$ proves this step.
- \circ Using our assumption that $f[V \cap A] \subseteq V'$ proves this step.
- \circ Using our assumption that V' is closed proves this step.

This completes the proof. \square

Expand all Collapse all

Current proof state:

X is a topological space
 Y is a regular topological space
 A is a dense subset of X (hA)
 $f : X \rightarrow Y$
For all elements x' of X , f is continuous at x' within A (hf)
 x is an element of X
 V' is a closed neighborhood of $f(x)$ (V'_in) (V'_closed)
 V is an open neighborhood of x (V_in) (V_op)
 $f[V \cap A] \subseteq V'$ (hV)
 z is an element of V (z_in)
 $f(z_1)$ tends to $f(z)$ as z_1 tends to z within $V \cap A$ (limV)

Goal: $f(z) \in V'$

(Example from a talk of Patrick Massot, 2023)

One notable feature of proof formalization projects is that they lend themselves to large collaborations that do not require high pre-established levels of trust. I recently led a project to formalize the following recent result, answering a question of Katalin Marton:

Polynomial Freiman–Ruzsa conjecture (Gowers, Green, Manners, T. 2023)

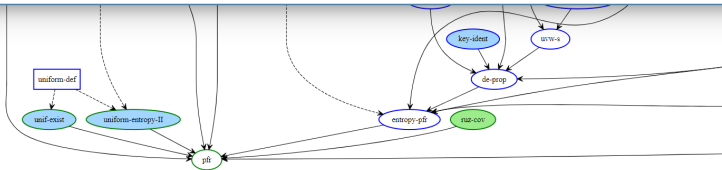
Let A be a finite subset of an elementary 2-group G such that $|A + A| \leq K|A|$. Then A can be covered by at most $2K^{12}$ cosets of a subgroup H of G of cardinality at most A .

This result was proven (by traditional means) in a 33-page, mostly self-contained paper. It was formalized in Lean in three weeks by a group of approximately twenty people.

Like the liquid tensor experiment, the formalization was organized through a human-readable blueprint, which broke up the main result into many intermediate propositions, the relationships between which can be visualized as a dependency graph.

(PFR)

\mathbf{F}_2^n and $|A + A| \leq K|A|$, then A can be covered by most $2K^{12}$ translates of a subspace H of \mathbf{F}_2^n with



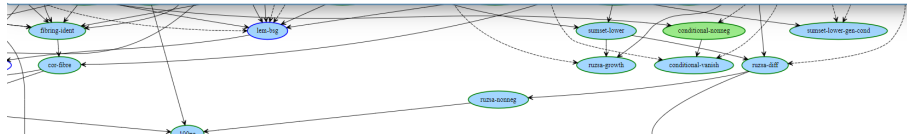
Each node in the graph can be formalized independently, using the predecessor results as “black boxes”. The process is quite parallelizable, and does not require high levels of pre-established trust between the participants, as the Lean compiler can be used to automatically verify contributions.

Lemma 3.13 (Distance nonnegative)

If X, Y are G -valued random variables, then

$$d[X; Y] \geq 0.$$

LaTeX Lean



This format allows for division of labor. A contributor who is expert in Lean can focus on formalizing one portion of the blueprint into Lean, without necessarily being a broader expert all the mathematical fields involved. Conversely, a mathematician who is not particularly familiar in Lean can work on the human-readable blueprint of the proof, leaving the technical formalization of the proof to others.

```

-   $$ d[X;Y] \geq 0. $$  -/
lemma rdist_nonneg : 0 ≤ d[ X ; μ # Y ; μ' ] := by
  suffices : 0 ≤ 2 * d[ X ; μ # Y ; μ' ]
  . linarith
have h : |H[X ; μ] - H[Y ; μ']| ≤ 2 * d[X ; μ # Y ; μ' ] :=
  exact diff_ent_le_rdist
have h' : 0 ≤ |H[X ; μ] - H[Y ; μ']| := by
  exact abs_nonneg (H[X ; μ] - H[Y ; μ'])
exact ge_trans h h'

```

- The formalization project revealed only minor typos in the human-generated proof, and generated many additions to Lean's Mathlib (e.g., the foundational theory of Shannon entropy and related aspects of probability theory).
- While the formal proof took longer generate than the human-readable proof, *modifying* the formal proof without generating errors takes about as much time as for a human-readable version. (For instance, we implemented a recent improvement of the exponent from 12 to 11 in a matter of days, and discovered a further numerical improvement of a related result in the course of the formalization process.)
- It seems possible to scale up the size of such projects to formalize more substantial results. For instance, Kevin Buzzard just launched a five-year project to formalize the entire proof of Fermat's last theorem in Lean, again relying on a blueprint to organize the project.

Both machine learning (ML) and non-ML tools are being used to close in on a notorious open problem in fluid equations - to demonstrate finite time blowup for the **incompressible Euler equations**.

- Numerical simulations (e.g., by Lou–Hou in 2014) suggested that there were solutions that exhibited self-similar behavior, but due to the finite accuracy of the simulations, the results were not rigorously conclusive.
- However, if one could use numerics to obtain *approximate* solutions that were very close to being self-similar, it would be possible in principle to perform a rigorous stability analysis (in self-similar coordinates) to then *prove* the existence of a solution that blew up in finite time.

- In 2022, there were two different approaches to generate approximate self-similar solutions; one (by Chen–Hou) using traditional numerical PDE methods, and a slightly earlier one (by Wang–Lai–Gómez-Serrano–Buckmaster) using a **Physics Informed Neural Network** (PINN).
- The Chen–Hou approach was also able to perform (with additional computer assistance) the rigorous stability analysis, establishing for the first time a rigorous finite time blowup result for smooth solutions to the incompressible Euler equations outside of a cylindrical boundary.
- There is now much activity using a variety of machine assisted methods to establish blowup for other fluid equations, though the most infamous problem of all - the Navier–Stokes global regularity problem - likely remains out of reach for now.

An interesting interaction between ML and human approaches to mathematics occurred in work by Alex Davies, András Juhász, Marc Lackenby, and Nenad Tomasev in 2021 in **knot theory**.



(“On Knots III”, P. Tait, 1885)

- Knot theory produces a wide array of **knot invariants** associated to a given knot: **signature**, **Jones polynomial**, **HOMFLY polynomial**, **hyperbolic invariants**, and so forth.
- These authors were seeking a connection between a knot's hyperbolic invariants (a collection of real and complex numbers) and its signature (an integer).

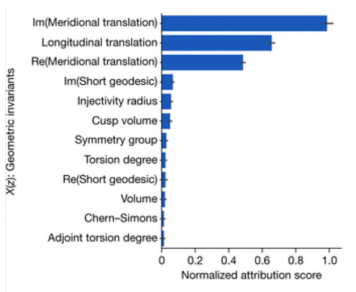
<i>link</i>	<i>volume</i>	<i>num sym</i>	<i>cusps vol</i>	<i>min trans</i>	<i>homological cusps shape</i>	<i>geometrical cusps shape</i>
L103001	(2,3) torus knot					
L104001	2.029883212819307	8	1.732051	1.000000	0.000000 + 0.288675 i	0.000000 + 0.288675 i
L105001	(2,5) torus knot					
L105002	2.828122088330783	4	1.973464	1.150964	0.165153 + 0.197597 i	0.053770 + 0.326785 i
L106001	3.163963228883144	4	1.995452	1.247409	-0.184237 + 0.258674 i	0.026219 + 0.388123 i
L106002	4.400832516123046	4	2.882878	1.283754	0.116833 + 0.060607 i	-0.020778 + 0.284311 i
L106003	5.693021091281301	8	4.038067	1.210608	0.000000 + 0.181469 i	0.000000 + 0.181469 i
L107001	(2,7) torus knot					
L107002	3.331744231641115	4	1.998818	1.300876	0.209974 + 0.238325 i	0.014545 + 0.422820 i
L107003	4.592125697027063	4	2.878339	1.340245	-0.102270 + 0.038195 i	0.040097 + 0.306790 i
L107004	5.137941201873418	8	3.538856	1.485116	-0.063371 + 0.298152 i	0.030574 + 0.308593 i
L107005	6.443537380850573	4	3.955450	1.222402	0.080637 + 0.045279 i	0.015187 + 0.187658 i
L107006	7.084925953510830	4	4.447335	1.226445	0.084410 + 0.089503 i	-0.012041 + 0.168247 i
L107007	7.643375172359955	8	5.065555	1.247409	0.038884 + 0.143017 i	-0.005414 + 0.153398 i
L108001	3.427205246274016	4	1.999601	1.332538	-0.219796 + 0.253214 i	0.008867 + 0.443826 i
L108002	4.935242678280654	4	2.761873	1.358235	0.084741 + 0.023099 i	-0.001737 + 0.333968 i
L108003	5.238684100798440	8	3.546273	1.526793	0.000000 + 0.328669 i	0.000000 + 0.328669 i
L108004	5.500486416347235	4	3.387435	1.509575	0.115347 + 0.045788 i	0.053907 + 0.327490 i
L108005	6.997189147792215	4	4.570602	1.605393	-0.081510 + 0.025954 i	-0.011036 + 0.281509 i

("Hyperbolic invariants of knots and links", C. Adams, M. Hildebrand, J. Weeks, 1991)

- By training a **neural network** on a database of over two million knots, they found that the network could predict the signature from the hyperbolic invariants with high accuracy.
- So there was a connection! And one which was not already predicted from existing theory.
- However, the relation produced by the neural net was a highly opaque “black box”; it was not clear from that network exactly what the relation was.

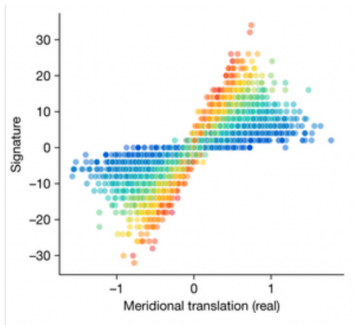
- But one could perform a **saliency analysis** - see how sensitive the neural net relation is to perturbation of one of its input parameters.
- This revealed that only three of the two dozen hyperbolic invariants had a significant effect on this relation.

The main hyperbolic invariants that were used to predict signature:



(“Using machine learning to formulate mathematical conjectures”, M. Lackenby, 2023)

- By visually inspecting scatterplots of the signature against these three invariants, they conjectured a more comprehensible relationship between these quantities.
- Further numerics disproved their initial conjecture, but suggested a modified version of the conjecture which they were able to prove rigorously.



A plot of signature against $\text{Re}(\mu)$ coloured by λ

Large language models

The recent emergence of powerful **Large Language Models**, such as GPT-4, shows promise. Here, GPT is able to solve a problem from the 2022 International Mathematical Olympiad:

GPT-4

Prompt:

Suppose g is a continuous smooth function such that for every $x > 0$, there is **one and only one** $y > 0$ such that $g(x) + g(y) \leq 2xy$. Can you prove that $g(x) = x^2$?

GPT-4:

There are a few possible ways to approach this problem, but one possible method is as follows:

- Assume, for the sake of contradiction, that $g(x)$ is not equal to x^2 for some $x > 0$.
- Then, either $g(x) < x^2$ or $g(x) > x^2$ for that x .
- If $g(x) < x^2$, then let $y = x$. Then $g(x) + g(y) = 2g(x) < 2x^2 = 2xy$.
- If $g(x) + g(y) < 2xy$, then $g(x) + g(y)$ is strictly less than the upper bound of $2xy$ for positive x and y .
- This means that there is some room to increase y slightly and still satisfy the inequality $g(x) + g(y) \leq 2xy$.
- For example, let $y' = y + \epsilon$, where ϵ is a small positive number such that $g(x) + g(y') < 2xy$. This is possible because g is continuous and smooth, so it does not have any jumps or kinks that would prevent such a choice of epsilon.
- Then, y' is another positive number such that $g(x) + g(y') \leq 2xy'$, which contradicts the assumption that y is the one and only one positive number that satisfies this condition given x .
- Therefore, if $g(x) < x^2$, then $g(x) + g(y)$ cannot be strictly less than $2xy$, and it must be equal to $2xy$.
- If $g(x) > x^2$, then let y^* be the unique positive solution of $g(x) + g(y) = 2xy$. Then $g(y^*) < (y^*)^2$, since otherwise we would have $g(x) + g(y^*) > 2xy$, which contradicts the existence of y^* given x .
- But then, by the same argument as before, we can let $x = y = y^*$ and obtain a contradiction, since $g(y) + g(y) = 2g(y) < 2y^2 = 2yy$, which contradicts the uniqueness of y given y .
- Therefore, $g(x)$ must be equal to x^2 for all $x > 0$, as required.

On the other hand, it can hallucinate mathematical nonsense, and even struggle with basic arithmetic:

$$7 * 4 + 8 * 8 =$$



120 [...]

$$7 * 4 + 8 * 8$$

= (7 * 4) + (8 * 8) (we can use parentheses to group multiplication terms, but they are not necessary)

= 28 + 64 (we perform the multiplication inside the parentheses first)

= 92 (we perform the addition last)

Wait but you started by saying $7 * 4 + 8 * 8 = 120$?



That was a typo, sorry. The correct answer is 92, as shown in the solution.

From a presentation of Bubeck and Eldan, 2023

- However, there are interesting experiments in boosting the power of large language models to do mathematics, either by connecting them with more traditional computational tools (such as Wolfram Alpha), by testing their output through a formal proof assistant and using that as feedback to reinforce the training of the model, or by combining them with more traditional mathematical problem solving strategies.
- Conversing with these models can also stimulate mathematical thought more indirectly, by suggesting “blue sky” proof approaches, locating related relevant concepts, or simply by being a patient listener to bounce ideas off of.

The Github Copilot tool is an AI that suggests potential code in various languages (including Lean) as an auto-complete, and has a moderate success rate in speeding up the process of writing code.

More advanced AI tools (e.g., Lean Copilot) are in development to suggest entire proofs of short mathematical statements, using proof assistants to filter out hallucinations and provide feedback to the AI.

```
;Y] \geq 0.$$ -/
```

```
nonneg : 0 ≤ d[ X ; μ # Y ; μ' ] := by  
0 ≤ 2 * d[ X ; μ # Y ; μ' ]
```

```
H[X ; μ] - H[Y ; μ'] | ≤ 2 * d[X ; μ # Y ; μ' ] := by  
of_nonneg (entropy_nonneg _), abs_of_nonneg (entropy_  
ff_ent_le_rdist
```

Where are we at now with machine assisted proofs?

- Computers by themselves still seem unlikely to resolve major mathematical problems on their own.
- However, they are increasingly being used to generate **assist** human mathematicians in a variety of creative ways, beyond just brute-force case checking or computation.
- For instance, we have seen they can be useful at generating conjectures or uncovering intriguing mathematical phenomena.
- Automated provers could also be used to explore the space of proofs itself, beyond the small set of “human-generatable” proofs that often require one to stay close to other sources of intuition, such as existing literature or connections to other ways of thinking.

- While AI technology shows great potential, in the immediate term, I expect it to have the most impact on tasks peripheral to mathematical research rather than central to it, such as automatically summarizing large amounts of literature or suggesting related work.
- Proof formalization continues to make steady improvements in speed and ease of use. The “de Bruijn factor” (the ratio between the difficulty of writing a correct formal proof and a correct informal proof) is still well above one (I estimate ~ 20), but dropping. Once AI integration takes place, this factor could potentially drop below one, which would be transformative to our field.

Thanks for listening!

