# AI in math and theoretical physics: status and prospects

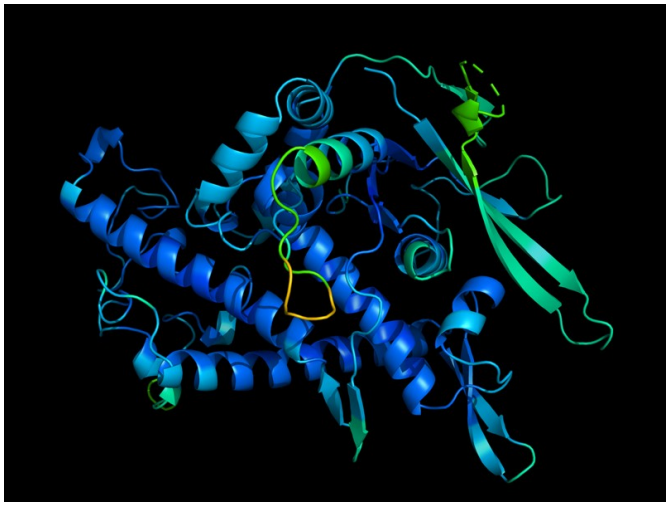
Michael R. Douglas

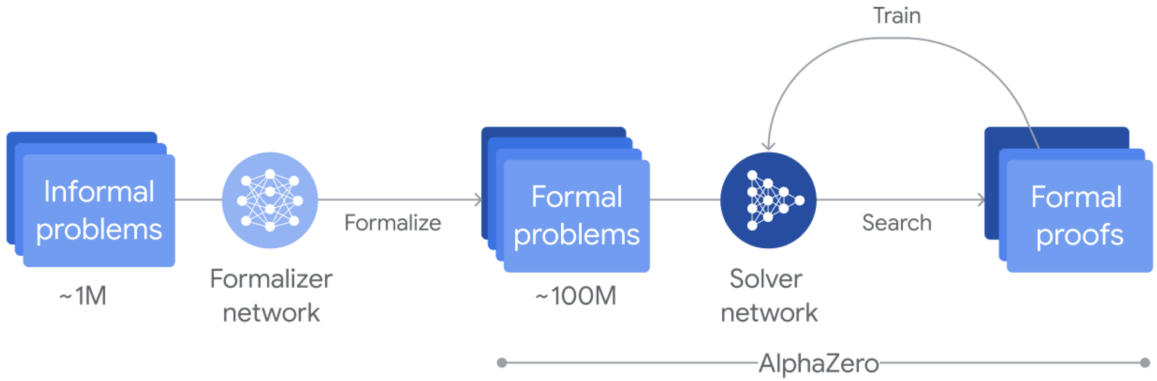
Mathematical Picture Language Seminar

## Abstract

AI progress is accelerating, and now the leaders expect “artificial general intelligence” (AGI) in 2 to 3 years. While difficult to believe, we must prepare for the possibility. We can take lessons from previous episodes in which computers surpassed humans, for example in chess. We discuss conceptual studies of mathematical copilots and autonomous mathematical discovery systems. These seem likely to me to become truly useful in the next few years.

# Machine learning and AI






Advancing faster than the experts predicted.

Artificial intelligence – methods for perception, reasoning, planning, decision making, representing knowledge which exhibit “intelligence.”
Machine learning – methods for learning from data, often statistical.
Definitions are somewhat time dependent, for example computer algebra was once considered AI.

Although the ideas developed over many decades, tangible results required advances in computing, especially (but not only) from Moore’s Law. See my recent work 2501.17894 with Sergiy Verstyuk, and the next slide. Since circa 2010, progress has been rapid and accelerating.

ML and AI have a large and growing impact on scientific research. They have been revolutionary in some fields, such as structural biology (AlphaFold), and they see widespread use in many fields, especially to analyze large datasets.

Exponential growth implies that the future impact can be far larger.

Artificial intelligence – methods for perception, reasoning, planning, decision making, representing knowledge which exhibit “intelligence.”
Machine learning – methods for learning from data, often statistical.
Definitions are somewhat time dependent, for example computer algebra was once considered AI.

Although the ideas developed over many decades, tangible results required advances in computing, especially (but not only) from Moore’s Law. See my recent work 2501.17894 with Sergiy Verstyuk, and the next slide. Since circa 2010, progress has been rapid and accelerating.

ML and AI have a large and growing impact on scientific research. They have been revolutionary in some fields, such as structural biology (AlphaFold), and they see widespread use in many fields, especially to analyze large datasets.

Exponential growth implies that the future impact can be far larger.

Artificial intelligence – methods for perception, reasoning, planning, decision making, representing knowledge which exhibit “intelligence.” Machine learning – methods for learning from data, often statistical. Definitions are somewhat time dependent, for example computer algebra was once considered AI.

Although the ideas developed over many decades, tangible results required advances in computing, especially (but not only) from Moore’s Law. See my recent work 2501.17894 with Sergiy Verstyuk, and the next slide. Since circa 2010, progress has been rapid and accelerating.

ML and AI have a large and growing impact on scientific research. They have been revolutionary in some fields, such as structural biology (AlphaFold), and they see widespread use in many fields, especially to analyze large datasets.

Exponential growth implies that the future impact can be far larger.

Artificial intelligence – methods for perception, reasoning, planning, decision making, representing knowledge which exhibit “intelligence.” Machine learning – methods for learning from data, often statistical. Definitions are somewhat time dependent, for example computer algebra was once considered AI.

Although the ideas developed over many decades, tangible results required advances in computing, especially (but not only) from Moore’s Law. See my recent work 2501.17894 with Sergiy Verstyuk, and the next slide. Since circa 2010, progress has been rapid and accelerating.

ML and AI have a large and growing impact on scientific research. They have been revolutionary in some fields, such as structural biology (AlphaFold), and they see widespread use in many fields, especially to analyze large datasets.

Exponential growth implies that the future impact can be far larger.

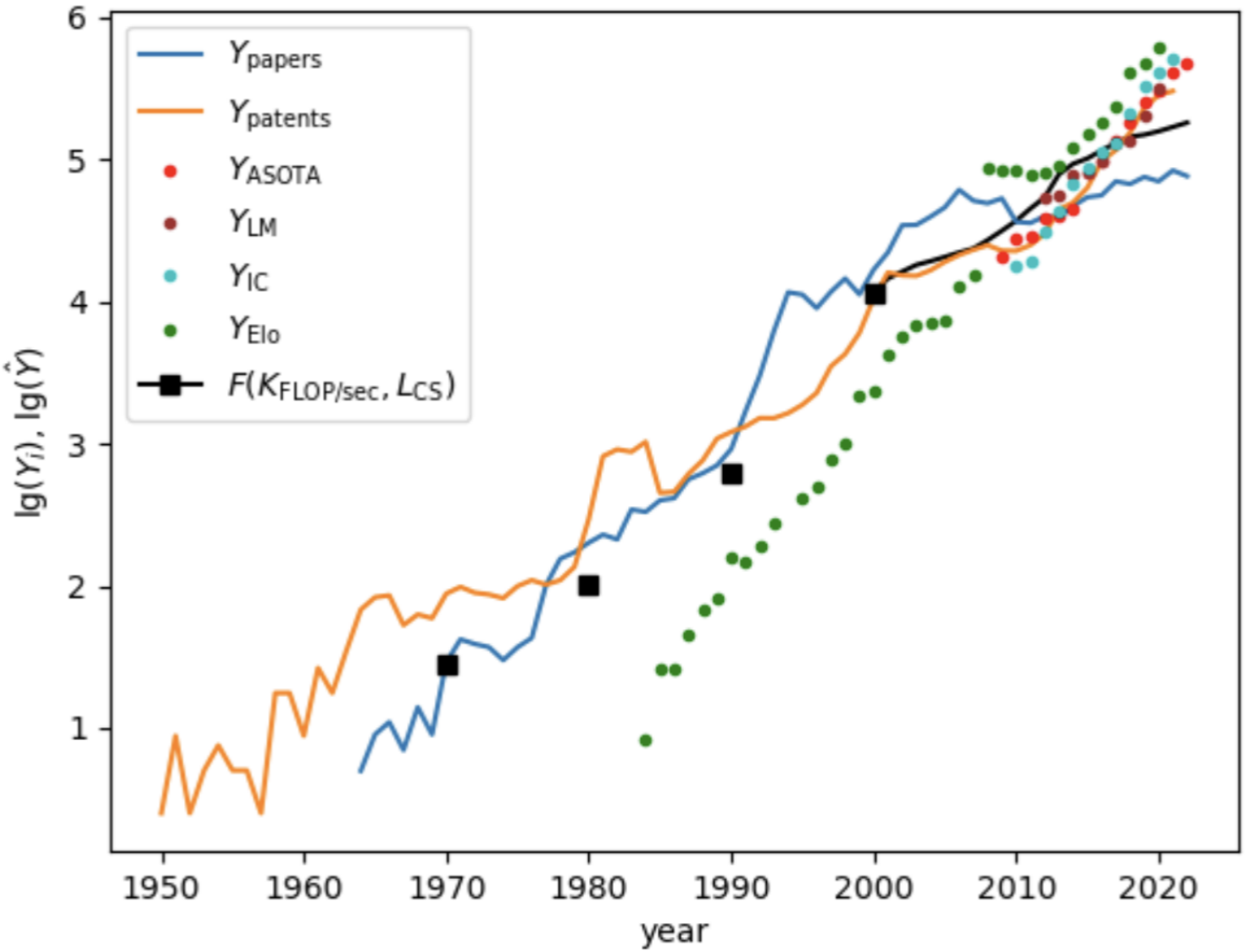



Figure 3: Progress in AI/ML technologies. From Douglas and Verstyuk, arXiv:2501.17894.

Admittedly, the impact on pure math and theoretical physics has been mild so far. I would guess that while many in this audience have dabbled with ML/AI, you don't use it in your regular work. (Do you?)

Some popular uses today:

- Finding patterns in data
- Better search engine (in multiple senses)
- Coding, code completion, simple textual manipulations
- Brainstorming

Some longstanding goals which have not yet materialized:

- AI research assistant
- Really good mathematical and scientific search engines (but see Deep Research below)
- Easy to use interactive theorem proving
- Autonomous mathematical discovery

Are there any fundamental barriers to achieving these goals? Probably not. But why are they not practical now, what would cause this to change, what is likely to change and on what timescale ?

Admittedly, the impact on pure math and theoretical physics has been mild so far. I would guess that while many in this audience have dabbled with ML/AI, you don't use it in your regular work. (Do you?)

Some popular uses today:

- Finding patterns in data
- Better search engine (in multiple senses)
- Coding, code completion, simple textual manipulations
- Brainstorming

Some longstanding goals which have not yet materialized:

- AI research assistant
- Really good mathematical and scientific search engines (but see Deep Research below)
- Easy to use interactive theorem proving
- Autonomous mathematical discovery

Are there any fundamental barriers to achieving these goals? Probably not. But why are they not practical now, what would cause this to change, what is likely to change and on what timescale ?

Admittedly, the impact on pure math and theoretical physics has been mild so far. I would guess that while many in this audience have dabbled with ML/AI, you don't use it in your regular work. (Do you?)

Some popular uses today:

- Finding patterns in data
- Better search engine (in multiple senses)
- Coding, code completion, simple textual manipulations
- Brainstorming

Some longstanding goals which have not yet materialized:

- AI research assistant
- Really good mathematical and scientific search engines (but see Deep Research below)
- Easy to use interactive theorem proving
- Autonomous mathematical discovery

Are there any fundamental barriers to achieving these goals? Probably not. But why are they not practical now, what would cause this to change, what is likely to change and on what timescale?

Admittedly, the impact on pure math and theoretical physics has been mild so far. I would guess that while many in this audience have dabbled with ML/AI, you don't use it in your regular work. (Do you?)

Some popular uses today:

- Finding patterns in data
- Better search engine (in multiple senses)
- Coding, code completion, simple textual manipulations
- Brainstorming

Some longstanding goals which have not yet materialized:

- AI research assistant
- Really good mathematical and scientific search engines (but see Deep Research below)
- Easy to use interactive theorem proving
- Autonomous mathematical discovery

Are there any fundamental barriers to achieving these goals? Probably not. But why are they not practical now, what would cause this to change, what is likely to change and on what timescale ?

From 2019–2021 I gave a talk at 30 physics and math departments and labs entitled “How will we do mathematics in 2030?” and including predictions. I spoke at the CMSA in February 2020 (v14). ST Yau was very enthusiastic and this is one reason I am here.

We will begin this talk by surveying the situation in 2025, including

- ML for scientific computation: PINN’s, subgrid modeling, *etc.*.
- Mathematical data science.
- Combinatorial optimization and search.
- Interactive theorem proving without and with AI.

Incremental so far – but progress is exponential and qualitative change can happen quickly. Consider computer chess – over the period 1994–2004 it went from low master ratings to superhuman ability. The analogy between then and now can be made quantitative as we explain later, and leads to the prediction that the period 2025–2035 will see a similar transition for research in the mathematical sciences. We will survey current thinking about what this will mean for us.

From 2019–2021 I gave a talk at 30 physics and math departments and labs entitled “How will we do mathematics in 2030?” and including predictions. I spoke at the CMSA in February 2020 (v14). ST Yau was very enthusiastic and this is one reason I am here.

We will begin this talk by surveying the situation in 2025, including

- ML for scientific computation: PINN’s, subgrid modeling, *etc.*.
- Mathematical data science.
- Combinatorial optimization and search.
- Interactive theorem proving without and with AI.

Incremental so far – but progress is exponential and qualitative change can happen quickly. Consider computer chess – over the period 1994–2004 it went from low master ratings to superhuman ability. The analogy between then and now can be made quantitative as we explain later, and leads to the prediction that the period 2025–2035 will see a similar transition for research in the mathematical sciences. We will survey current thinking about what this will mean for us.

From 2019–2021 I gave a talk at 30 physics and math departments and labs entitled “How will we do mathematics in 2030?” and including predictions. I spoke at the CMSA in February 2020 (v14). ST Yau was very enthusiastic and this is one reason I am here.

We will begin this talk by surveying the situation in 2025, including

- ML for scientific computation: PINN’s, subgrid modeling, *etc.*.
- Mathematical data science.
- Combinatorial optimization and search.
- Interactive theorem proving without and with AI.

Incremental so far – but progress is exponential and qualitative change can happen quickly. Consider computer chess – over the period 1994–2004 it went from low master ratings to superhuman ability. The analogy between then and now can be made quantitative as we explain later, and leads to the prediction that the period 2025–2035 will see a similar transition for research in the mathematical sciences. We will survey current thinking about what this will mean for us.

From 2019–2021 I gave a talk at 30 physics and math departments and labs entitled “How will we do mathematics in 2030?” and including predictions. I spoke at the CMSA in February 2020 (v14). ST Yau was very enthusiastic and this is one reason I am here.

We will begin this talk by surveying the situation in 2025, including

- ML for scientific computation: PINN’s, subgrid modeling, *etc.*.
- Mathematical data science.
- Combinatorial optimization and search.
- Interactive theorem proving without and with AI.

Incremental so far – but progress is exponential and qualitative change can happen quickly. Consider computer chess – over the period 1994–2004 it went from low master ratings to superhuman ability. The analogy between then and now can be made quantitative as we explain later, and leads to the prediction that the period 2025–2035 will see a similar transition for research in the mathematical sciences. We will survey current thinking about what this will mean for us.

# Two slide review of machine learning

ML combines methods from statistics, applied math (numerical methods, optimization) and computer science and also takes ideas from neuroscience, physics, and cognitive science.
Models: parameterized functions $F : \mathcal{D}_{in} \times \mathcal{W} \to \mathcal{D}_{out}$. $\mathcal{D}_{in} \cong \mathbb{R}^D$ inputs, $\mathcal{D}_{out}$ outputs. $\mathcal{W} \cong \mathbb{R}^{N_p}$ weights (parameters).
Feed-forward neural networks (FFNs), transformers, decision trees...

$$F(x; \Theta_0, \Theta_1, \ldots, \Theta_L) = \Theta_L \circ \sigma \circ \Theta_{L-1} \circ \sigma \circ \ldots \circ \sigma \circ \Theta_0 \circ \vec{x}.$$

Three broad paradigms:

- Supervised learning – learn a function $f : \mathcal{D}_{in} \to \mathcal{D}_{out}$.
- Self-supervised learning (or semi- or un-) – learn a probability distribution on $\mathcal{D}_{in}$.
- Reinforcement learning – learn "to play a game," often with infrequent rewards.

Model fitting: optimization of an objective (loss) function, usually by gradient descent.

# Two slide review of machine learning

ML combines methods from statistics, applied math (numerical methods, optimization) and computer science and also takes ideas from neuroscience, physics, and cognitive science.
Models: parameterized functions $F : \mathcal{D}_{in} \times \mathcal{W} \to \mathcal{D}_{out}$. $\mathcal{D}_{in} \cong \mathbb{R}^D$ inputs, $\mathcal{D}_{out}$ outputs. $\mathcal{W} \cong \mathbb{R}^{N_p}$ weights (parameters).
Feed-forward neural networks (FFNs), transformers, decision trees...

$$F(x; \Theta_0, \Theta_1, \ldots, \Theta_L) = \Theta_L \circ \sigma \circ \Theta_{L-1} \circ \sigma \circ \ldots \circ \sigma \circ \Theta_0 \circ \vec{x}.$$

Three broad paradigms:

- Supervised learning – learn a function $f : \mathcal{D}_{in} \to \mathcal{D}_{out}$.
- Self-supervised learning (or semi- or un-) – learn a probability distribution on $\mathcal{D}_{in}$.
- Reinforcement learning – learn "to play a game," often with infrequent rewards.

Model fitting: optimization of an objective (loss) function, usually by gradient descent.

Just as important as the hardware and software are ML concepts:

- Importance of datasets and data curation.
- Generalization and inductive bias.
- Training, validation and testing to judge generalization.
- Benign overparameterization (instead of overfitting).
- Lore about which models and methods work for which problems.

The flip side of the generality of ML is difficulty of interpretability. A neural network might be able to approximate a function $y = f(x)$ with 99% accuracy, but still give little insight which humans can understand. Simpler models (especially linear) are preferable.

Interpretability techniques: attribution analysis (which data features and which model components drive predictions), distillation (to get smaller models), symbolic regression, computational models.

Just as important as the hardware and software are ML concepts:

- Importance of datasets and data curation.
- Generalization and inductive bias.
- Training, validation and testing to judge generalization.
- Benign overparameterization (instead of overfitting).
- Lore about which models and methods work for which problems.

The flip side of the generality of ML is difficulty of interpretability. A neural network might be able to approximate a function $y = f(x)$ with 99% accuracy, but still give little insight which humans can understand. Simpler models (especially linear) are preferable.

Interpretability techniques: attribution analysis (which data features and which model components drive predictions), distillation (to get smaller models), symbolic regression, computational models.

Just as important as the hardware and software are ML concepts:

- Importance of datasets and data curation.
- Generalization and inductive bias.
- Training, validation and testing to judge generalization.
- Benign overparameterization (instead of overfitting).
- Lore about which models and methods work for which problems.

The flip side of the generality of ML is difficulty of interpretability. A neural network might be able to approximate a function $y = f(x)$ with 99% accuracy, but still give little insight which humans can understand. Simpler models (especially linear) are preferable.

Interpretability techniques: attribution analysis (which data features and which model components drive predictions), distillation (to get smaller models), symbolic regression, computational models.

# ML for scientific computation

This is probably the most widely known of the topics and I won't try to cover it in any depth. Core ideas:

- Replace standard numerical tools with ML models, especially representing, fitting, interpolating and extrapolating functions.
- Reformulate solution methods as optimization, *e.g.* solve $E = 0 \Rightarrow$ minimize $|E|$ or $|E|^2$.
- Leverage ML hardware/software: GPUs, CUDA, PyTorch, Jax...
- Accept imprecise, statistically good results. Rigorous ML is hard.

Some general methods:

- Distillation – take a large dataset of simulations, or a large model, and extract simpler models and/or summary results using ML.
- Subgrid modeling. Numerical representations of functions, such as discretization, have a "resolution" or smallest length scale. Instead of accepting the resulting error, try to model the system (usually a PDE) on smaller scales using ML.

# ML for scientific computation

This is probably the most widely known of the topics and I won't try to cover it in any depth. Core ideas:

- Replace standard numerical tools with ML models, especially representing, fitting, interpolating and extrapolating functions.
- Reformulate solution methods as optimization, *e.g.* solve $E = 0 \Rightarrow$ minimize $|E|$ or $|E|^2$.
- Leverage ML hardware/software: GPUs, CUDA, PyTorch, Jax...
- Accept imprecise, statistically good results. Rigorous ML is hard.

Some general methods:

- Distillation – take a large dataset of simulations, or a large model, and extract simpler models and/or summary results using ML.
- Subgrid modeling. Numerical representations of functions, such as discretization, have a "resolution" or smallest length scale. Instead of accepting the resulting error, try to model the system (usually a PDE) on smaller scales using ML.

A few noteworthy examples:

- Hybrid weather and climate models – replace some model components with ML, including subgrid modeling. Much reduced compute means one can generate many more forecasts and thus deal with uncertainty.
- More generally, fitting simulations (synthetic data).
- More efficient lattice gauge theory by optimizing sampling (Cranmer *et al* 2309.01156).
- Modeling in neuroscience (a prime focus at Kempner).
- Mathematical understanding of fluid dynamics, Navier-Stokes problem. See recent works of Chen and Hou, and Buckmaster and Gomez-Serrano, and others on the search for singular solutions to Navier-Stokes.

# Calabi-Yau and $G_2$ holonomy metrics

A string theory application is to the Ricci-flat Kähler metrics on Calabi-Yau manifolds. While these are proven to exist, explicit expressions are believed not to exist. Starting in 2020 we and others used ML techniques to get approximate metrics on general CY manifolds (no symmetry) with better than 1% accuracy.

In 2405.19402 with Platt, Qi and Barbosa we found a numerical $G_2$ holonomy metric using the construction of Joyce and Karagiannis (2017). This starts from CY $\times S^1/\mathbb{Z}_2$ where the $\mathbb{Z}_2$ fixes a special Lagrangian submanifold $L$. J and K show that these singularities can be resolved iff $L$ has a nonvanishing harmonic 1-form. This is a non-topological condition, but one we can check numerically.

With Gomez-Serrano, Lehmann, Tong, Platt and Qi, we are working on a **rigorous** proof that a particular approximate CY metric has a nearby exact CY metric. We use a fixed point theorem and replace the *a priori* estimates with verified numerical computations.

# Calabi-Yau and $G_2$ holonomy metrics

A string theory application is to the Ricci-flat Kähler metrics on Calabi-Yau manifolds. While these are proven to exist, explicit expressions are believed not to exist. Starting in 2020 we and others used ML techniques to get approximate metrics on general CY manifolds (no symmetry) with better than 1% accuracy.

In 2405.19402 with Platt, Qi and Barbosa we found a numerical $G_2$ holonomy metric using the construction of Joyce and Karagiannis (2017). This starts from CY $\times$ $S^1/\mathbb{Z}_2$ where the $\mathbb{Z}_2$ fixes a special Lagrangian submanifold $L$. J and K show that these singularities can be resolved iff $L$ has a nonvanishing harmonic 1-form. This is a non-topological condition, but one we can check numerically.

With Gomez-Serrano, Lehmann, Tong, Platt and Qi, we are working on a **rigorous** proof that a particular approximate CY metric has a nearby exact CY metric. We use a fixed point theorem and replace the *a priori* estimates with verified numerical computations.

# Calabi-Yau and $G_2$ holonomy metrics

A string theory application is to the Ricci-flat Kähler metrics on Calabi-Yau manifolds. While these are proven to exist, explicit expressions are believed not to exist. Starting in 2020 we and others used ML techniques to get approximate metrics on general CY manifolds (no symmetry) with better than 1% accuracy.

In 2405.19402 with Platt, Qi and Barbosa we found a numerical $G_2$ holonomy metric using the construction of Joyce and Karagiannis (2017). This starts from CY $\times$ $S^1/\mathbb{Z}_2$ where the $\mathbb{Z}_2$ fixes a special Lagrangian submanifold $L$. J and K show that these singularities can be resolved iff $L$ has a nonvanishing harmonic 1-form. This is a non-topological condition, but one we can check numerically.

With Gomez-Serrano, Lehmann, Tong, Platt and Qi, we are working on a **rigorous** proof that a particular approximate CY metric has a nearby exact CY metric. We use a fixed point theorem and replace the *a priori* estimates with verified numerical computations.

# Mathematical data science

A mathematical (or platonic) data set is (Douglas and Lee 2502.08620)

- A set $\mathcal{S}$ of mathematical objects.
- A function $F$ on $\mathcal{S}$ into a space of invariants $\mathcal{D}$, say $\mathbb{R}^d$.
- A way to pick out finite subsets $\mathcal{S}_i \subseteq \mathcal{S}$ – this could be filtration by size parameter, or sampling from a probability distribution $\rho$ on $\mathcal{S}$.

Examples:

- Number theory: LMFDB of curves and L-functions, modular forms
- Knot theory: Regina knot census of knots up to 19 crossings
- Algebraic geometry: Kreutzer-Skarke DB of reflexive polytopes
- Graph theory: usually custom, though see "House of Graphs."

ML generally doesn't know any math, it looks for patterns in the point clouds $F(\mathcal{S}_i)$. What kind of patterns can one find using it?

# Mathematical data science

A mathematical (or platonic) data set is (Douglas and Lee 2502.08620)

- A set $\mathcal{S}$ of mathematical objects.
- A function $F$ on $\mathcal{S}$ into a space of invariants $\mathcal{D}$, say $\mathbb{R}^d$.
- A way to pick out finite subsets $\mathcal{S}_i \subseteq \mathcal{S}$ – this could be filtration by size parameter, or sampling from a probability distribution $\rho$ on $\mathcal{S}$.

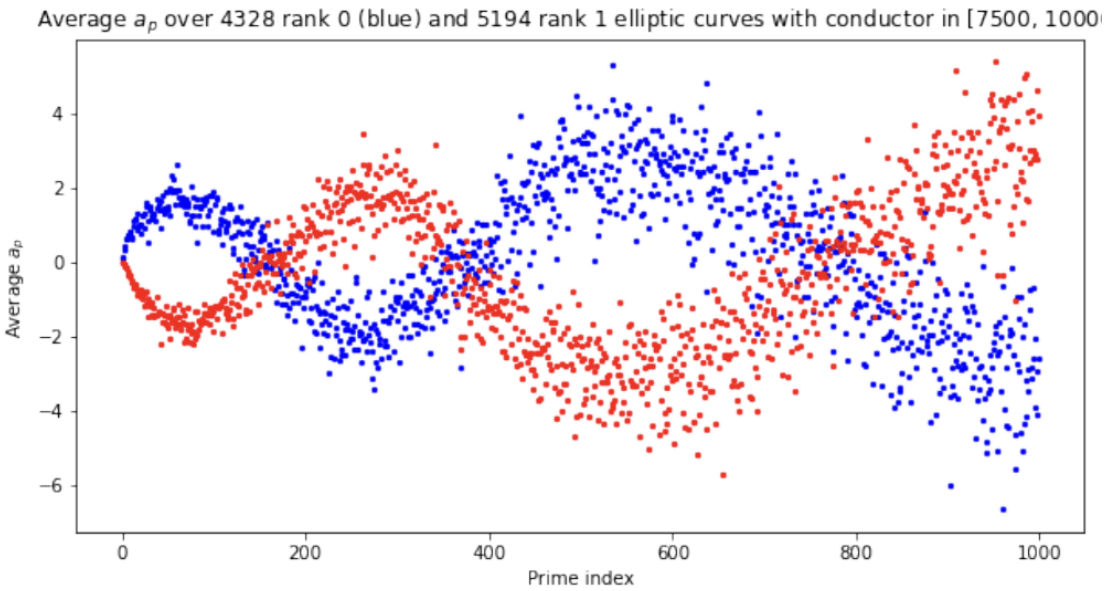
Examples:

- Number theory: LMFDB of curves and L-functions, modular forms
- Knot theory: Regina knot census of knots up to 19 crossings
- Algebraic geometry: Kreutzer-Skarke DB of reflexive polytopes
- Graph theory: usually custom, though see “House of Graphs.”

ML generally doesn’t know any math, it looks for patterns in the point clouds $F(\mathcal{S}_i)$. What kind of patterns can one find using it?

# Murmurations

The L-function of a curve $E$ has coefficients $a_p(E)$ derived from the number of its solutions in a finite field $\mathbb{F}_p$.




Averaging $a_p$ over large numbers of curves with nearby conductor reveals unexpected oscillatory patterns. From He *et al* 2204.10140.

$$f_r(n) = \frac{1}{\#\mathcal{E}_r[N_1, N_2]} \sum_{E \in \mathcal{E}_r[N_1, N_2]} a_{p_n}(E),$$

where $\mathcal{E}_r[N_1, N_2]$ is the set of elliptic curves over $\mathbb{Q}$ with conductor in range $[N_1, N_2]$, and rank 0 for blue, rank 1 for red.

# Learning Fricke signs using LDA from Maass Form Coefficients

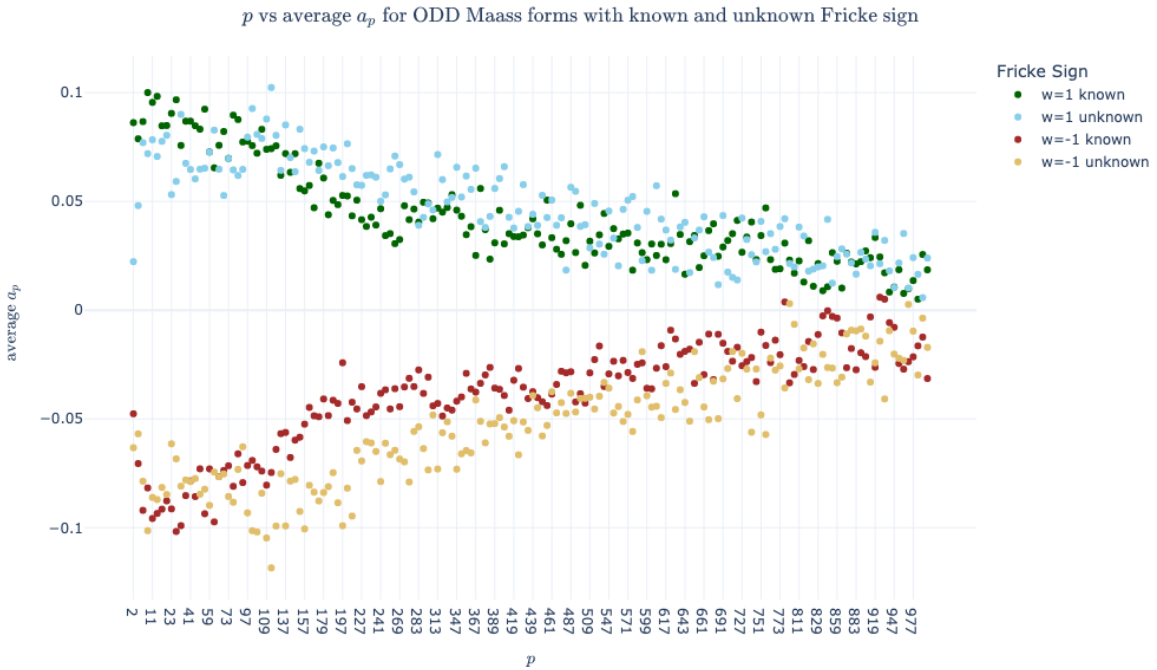
Joanna Bieri, Giorgi Butbaia, Edgar Costa, Alyson Deines, Kyu-Hwan Lee, David Lowry-Duda, Tom Oliver, Yidi Qi, and Tamara Veenstra

35K Maass forms in LMFDB: Fricke signs missing for about half.

- Can we predict with LDA?
- How do predictions compare to Hejhal's heuristic algorithm?

| | |
|---|---|
| Accuracy | 96.12% |
| Heuristic | 95.45% |

## Murmurations of known versus learned Fricke signs








From Bieri, Butbaia, Costa, Deines, Lee, Lowry-Duda, Oliver, Qi and Veenstra, arXiv:2501.02105, done at the CMSA Math-ML workshop.

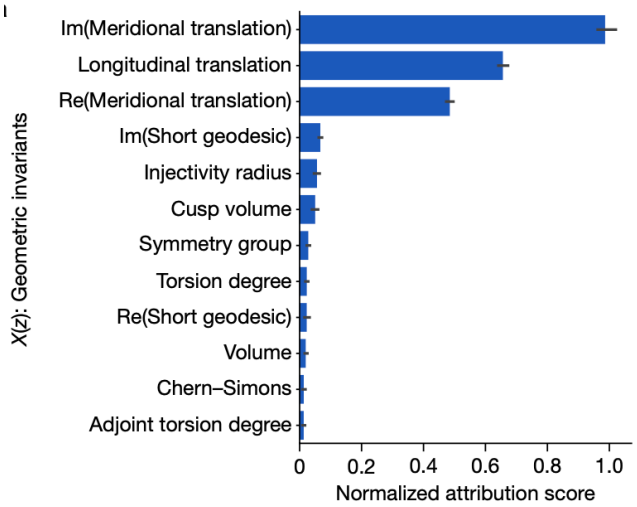
# Knot theory – relation between invariants

| $z$: **Knot** | $X(z)$: **Geometric invariants** | | | | $Y(z)$: **Algebraic invariants** | | |
|---|---|---|---|---|---|---|---|
| | Volume | Chern–Simons | Meridional translation | ... | Signature | Jones polynomial | ... |
|  | 2.0299 | 0 | $i$ | ... | 0 | $t^{-2} - t^{-1} + 1 - t + t^2$ | ... |
|  | 2.8281 | −0.1532 | $0.7381 + 0.8831i$ | ... | −2 | $t - t^2 + 2t^3 - t^4 + t^5 - t^6$ | ... |
|  | 3.1640 | 0.1560 | $-0.7237 + 1.0160i$ | ... | 0 | $t^{-2} - t^{-1} + 2 - 2t + t^2 - t^3 + t^4$ | ... |

## Theorem

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

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




From Davies *et al* (2021).

# Constrained optimization and search

Constraint satisfaction and constrained optimization are core CS topics, for which ML is just one class of methods among many. The archetypal problem is satisfiability or *SAT*. One is given a Boolean expression $S$ (the "constraint") in $n$ variables $x_1, x_2, \ldots, x_n$, say $S = x_1$ AND ($x_2$ OR NOT $x_3$). The problem is to find an assignment of truth values to variables for which $S$ is true, or prove that none exists.

Finding an assignment is in NP (a solution can be efficiently verified), while proving none exists is in co-NP. Loosely speaking, there is believed to be no general method (for either) much more efficient than "brute force" search. One can also count solutions, the #*SAT* problem. Or given $m$ clauses, find the variable assignment which maximizes the number which are true (*MAX-SAT*).

One can generalize to multivalued variables, and more general constraints such as $x_1 + x_2 = x_3$ in $\mathbb{Z}_p$.

# Constrained optimization and search

Constraint satisfaction and constrained optimization are core CS topics, for which ML is just one class of methods among many. The archetypal problem is satisfiability or *SAT*. One is given a Boolean expression $S$ (the "constraint") in $n$ variables $x_1, x_2, \ldots, x_n$, say $S = x_1$ AND $(x_2$ OR NOT $x_3)$. The problem is to find an assignment of truth values to variables for which $S$ is true, or prove that none exists.

Finding an assignment is in NP (a solution can be efficiently verified), while proving none exists is in co-NP. Loosely speaking, there is believed to be no general method (for either) much more efficient than "brute force" search. One can also count solutions, the *#SAT* problem. Or given $m$ clauses, find the variable assignment which maximizes the number which are true (*MAX-SAT*).

One can generalize to multivalued variables, and more general constraints such as $x_1 + x_2 = x_3$ in $\mathbb{Z}_p$.

In constrained optimization one also has an objective function to optimize. Some math problems naturally fit into this framework, especially those coming from extremal combinatorics:

- How many edges can a graph have, with $n$ vertices and no 4-cycles?
- The cap set problem: what is the largest set of vectors $v_i \in \mathbb{Z}_3^n$ such that $v_i + v_j + v_k \neq 0 \; \forall i \neq j \neq k$ ?
- The empty hexagon number: what is the largest number $n$ of points $x_i \in \mathbb{R}^2$ in general position (no three on a line), such that there must exist a convex hexagon containing no points?
- What is the maximal number of edges one can delete from the d-dimensional hypercube, without increasing its diameter?

A more general class of problems is combinatorial optimization – there is a discrete configuration space and an objective function, but the space is not defined by constraints. For example, pathfinding – given two vertices in a graph, find a connecting path. Taking a Cayley graph, one gets many mathematically important problems.

In constrained optimization one also has an objective function to optimize. Some math problems naturally fit into this framework, especially those coming from extremal combinatorics:

- How many edges can a graph have, with $n$ vertices and no 4-cycles?
- The cap set problem: what is the largest set of vectors $v_i \in \mathbb{Z}_3^n$ such that $v_i + v_j + v_k \neq 0 \; \forall i \neq j \neq k$ ?
- The empty hexagon number: what is the largest number $n$ of points $x_i \in \mathbb{R}^2$ in general position (no three on a line), such that there must exist a convex hexagon containing no points?
- What is the maximal number of edges one can delete from the d-dimensional hypercube, without increasing its diameter?

A more general class of problems is combinatorial optimization – there is a discrete configuration space and an objective function, but the space is not defined by constraints. For example, pathfinding – given two vertices in a graph, find a connecting path. Taking a Cayley graph, one gets many mathematically important problems.

Methods and concepts:

- Local search – given a configuration, consider local actions (*e.g.* add/remove edges from a graph) and "climb".
- Backtracking search – keep track of previous configurations and allow returning to them. Many methods: breadth-first, depth-first, beam search, $A^*$...
- Constraint propagation. For example, suppose $S$ includes the clause $x_1$ OR NOT $x_2$ and we know $x_2 = T$, then we can infer $x_1 = T$. A more sophisticated method is CDCL, which allows carrying information to other parts of the search tree.
- Genetic algorithms and generative search. These are ways to move farther in the configuration space, helping to jump out of local minima.

All of these involve heuristics (rules for which actions to take), value functions to optimize, *etc.* and ML is generally applied to learn these. Some of these problems (especially *SAT*) have received intense effort, and the best solvers don't use ML. But ML has had successes, most famously adversarial search (chess, go).

# FunSearch

An example of AI used for combinatorial optimization is FunSearch (Romera-Paredes *et al* 2024). It searches for (Python) programs which solve a problem, by generating them with a pre-trained LLM.

- LLM prompt = evaluation function + a list of programs $P_1, P_2$, ... which each generate solution, followed by `def P_k:`
- LLM output = new program $P_k$ incorporating features of $P_1, P_2$, ...

This is used in a genetic algorithm – the state is a population of correct and highly scoring programs; new programs are evaluated and may be added to the population.

Used with Ellenberg *et al* to find new solutions to the cap set problem: largest subset $S \subset \mathbb{Z}_3^d$ such that no triple $a, b, c \in S$ satisfies $a + b + c = 0$.

The program $P$ does not directly generate $S$, rather it defines a priority function $\rho : \mathbb{Z}_3^d \to \mathbb{R}$. $S$ is then generated by greedily taking each argmax $\rho$ which satisfies the constraints. In this sense the algorithm uses ML as a heuristic for combinatorial optimization.

# FunSearch applied to the cap set problem

Priority function $\rho : \mathbb{Z}_3^d \to \mathbb{R}$ for $d = 8$. Generates a cap set of size 512, beating the old record.

While the definition is not completely transparent, it is far more so than a set of FFN weights. By studying it, the authors found a simpler explicit construction of the same cap set.

Is the LLM applying problem-specific heuristics? If so, why did it work? Because there are good heuristics in the corpus? How could we test this hypothesis? It would be interesting to compare with a similar genetic search which did not have access to the NL corpus.

```
def priority(el: tuple[int,...],
↪  n: int) -> float:
  score = n
  in_el = 0
  el_count = el.count(0)

  if el_count == 0:
    score += n**2
    if el[1] == el[-1]:
      score *= 1.5
    if el[2] == el[-2]:
      score *= 1.5
    if el[3] == el[-3]:
      score *= 1.5
  else:
    if el[1] == el[-1]:
      score *= 0.5
    if el[2] == el[-2]:
      score *= 0.5

  for e in el:
    if e == 0:
      if in_el == 0:
        score *= n * 0.5
      elif in_el == el_count - 1:
        score *= 0.5
      else:
        score *= n * 0.5 ** in_el
      in_el += 1
    else:
      score += 1

  if el[1] == el[-1]:
    score *= 1.5
  if el[2] == el[-2]:
    score *= 1.5

  return score
```

# Interactive theorem proving

Expressing mathematical proof in a completely explicit and verifiable way is an ancient dream (Leibniz, Hilbert, ...). The concept of formal proof was developed in the early 20th century, but it soon emerged (as in Russell and Whitehead's *Principia Mathematica*) that formal proofs of serious mathematics are very long and difficult to write.

Solving this problem required developments both in logic (dependent type theory) and computer science (software engineering practices, object oriented programming languages, *etc.*). By the mid-2000's serious mathematical theorems were being verified by Gonthier's INRIA group using Coq (the four color theorem, the Feit-Thompson theorem). Hales' Flyspeck project verified his solution of the Kepler problem using HOL. Immler's 2017 verification of Tucker's proof that the Lorenz system is a strange attractor was a noteworthy example of verified numerical computation.

# Interactive theorem proving

Expressing mathematical proof in a completely explicit and verifiable way is an ancient dream (Leibniz, Hilbert, ...). The concept of formal proof was developed in the early 20th century, but it soon emerged (as in Russell and Whitehead's *Principia Mathematica*) that formal proofs of serious mathematics are very long and difficult to write.

Solving this problem required developments both in logic (dependent type theory) and computer science (software engineering practices, object oriented programming languages, *etc.*). By the mid-2000's serious mathematical theorems were being verified by Gonthier's INRIA group using Coq (the four color theorem, the Feit-Thompson theorem). Hales' Flyspeck project verified his solution of the Kepler problem using HOL. Immler's 2017 verification of Tucker's proof that the Lorenz system is a strange attractor was a noteworthy example of verified numerical computation.

The SOTA in ITP for math is arguably Lean 4 (De Moura *et al*) though the Isabelle and Coq systems also have strong points. A sizable community of mathematicians (100's) is using Lean, and its library of proven theorems (Lean mathlib) has about 1.7M lines of code. It covers much of the (French) undergrad math curriculum, and more advanced results esp. in number theory, category theory, abstract algebra, measure theory, multivariate calculus, manifolds.

Some recent accomplishments:

- Verification of Scholze and Clausen's "liquid mathematics."
- The independence of the continuum hypothesis (Han and van Doorn 1904.10570), polynomial Freiman-Ruzsa conjecture (Gowers *et al* 2311.05762), ...
- In progress: Fermat-Wiles (Buzzard), sphere eversion (Massot).

But, Lean is hard to learn ($\sim$ 1 year to become proficient) and the proofs are not easy to read. The "assembly language of proof."

The SOTA in ITP for math is arguably Lean 4 (De Moura *et al*) though the Isabelle and Coq systems also have strong points. A sizable community of mathematicians (100's) is using Lean, and its library of proven theorems (Lean mathlib) has about 1.7M lines of code. It covers much of the (French) undergrad math curriculum, and more advanced results esp. in number theory, category theory, abstract algebra, measure theory, multivariate calculus, manifolds.

Some recent accomplishments:

- Verification of Scholze and Clausen's "liquid mathematics."
- The independence of the continuum hypothesis (Han and van Doorn 1904.10570), polynomial Freiman-Ruzsa conjecture (Gowers *et al* 2311.05762), ...
- In progress: Fermat-Wiles (Buzzard), sphere eversion (Massot).

But, Lean is hard to learn ($\sim$ 1 year to become proficient) and the proofs are not easy to read. The "assembly language of proof."

```
/-- **Pythagorean theorem**, if-and-only-if angle-at-point form. -/
theorem dist_sq_eq_dist_sq_add_dist_sq_iff_angle_eq_pi_div_two (p1 p2 p3 : P) :
    dist p1 p3 * dist p1 p3 = dist p1 p2 * dist p1 p2 + dist p3 p2 * dist p3 p2 ↔
      ∠ p1 p2 p3 = π / 2 := by
  erw [dist_comm p3 p2, dist_eq_norm_vsub V p1 p3, dist_eq_norm_vsub V p1 p2,
    dist_eq_norm_vsub V p2 p3, ← norm_sub_sq_eq_norm_sq_add_norm_sq_iff_angle_eq_pi_div_two,
    vsub_sub_vsub_cancel_right p1, ← neg_vsub_eq_vsub_rev p2 p3, norm_neg]
```

```
/-- An angle in a right-angled triangle expressed using `arccos`. -/
theorem angle_eq_arccos_of_angle_eq_pi_div_two {p₁ p₂ p₃ : P} (h : ∠ p₁ p₂ p₃ = π / 2) :
    ∠ p₂ p₃ p₁ = Real.arccos (dist p₃ p₂ / dist p₁ p₃) := by
  rw [angle, ← inner_eq_zero_iff_angle_eq_pi_div_two, real_inner_comm, ← neg_eq_zero, ←
    inner_neg_left, neg_vsub_eq_vsub_rev] at h
  rw [angle, dist_eq_norm_vsub' V p₃ p₂, dist_eq_norm_vsub V p₁ p₃, ← vsub_add_vsub_cancel p₁ p₂
    add_comm, angle_add_eq_arccos_of_inner_eq_zero h]
```

The proofs (from Lean mathlib) of the Pythagorean theorem, and of the expression $\cos\theta = |p_3 - p_2|/|p_3 - p_1|$ for the angle at $p_3$ in a right triangle.

# AI for theorem proving

Automated theorem proving and increased automation in interactive theorem proving have a long history in computer science/formal methods. Logics are generally divided into propositional, first order (quantification over sets but not functions), and higher order logic (quantification over functions). Much of mathematics can only be defined in HOL, for example the field of real numbers. However, automated proof for propositional logic (SAT solvers) and first order logic is much more highly developed than for HOL.

This leads to the "sledgehammer" concept: to prove X in HOL, (1) find a small set of prerequisite lemmas, (2) translate X and the lemmas into FOL, (3) use a FOL prover, (4) translate back. This was developed for Isabelle by Paulson *et al*, works well, and makes proofs much easier to read. It is being developed for Lean at the Hoskinson Center at CMU.

# AI for theorem proving

Automated theorem proving and increased automation in interactive theorem proving have a long history in computer science/formal methods. Logics are generally divided into propositional, first order (quantification over sets but not functions), and higher order logic (quantification over functions). Much of mathematics can only be defined in HOL, for example the field of real numbers. However, automated proof for propositional logic (SAT solvers) and first order logic is much more highly developed than for HOL.

This leads to the "sledgehammer" concept: to prove X in HOL, (1) find a small set of prerequisite lemmas, (2) translate X and the lemmas into FOL, (3) use a FOL prover, (4) translate back. This was developed for Isabelle by Paulson *et al*, works well, and makes proofs much easier to read. It is being developed for Lean at the Hoskinson Center at CMU.

The use of ML in automated proving is still experimental but shows steady progress. One paradigm (almost universal until recently) is reinforcement learning (RL). A prover must make choices along the way: choice of previously proven results to use (premise selection), hypotheses to consider (such as "set $x =$ TRUE" in *SAT*), and choice of rewriting rules and other "tactics" to apply. Many systems follow hand-coded heuristic rules to make these choices.

But in RL, the system learns to make these choices, most famously as done by AlphaZero. The choices are actions, like moves in a game of solitaire. A complete proof gets a reward (winning the game).

Early works on HOL proving following this approach include TacticToe (Gauthier, Kalisczyk and Urban 2017), GamePad (Huang, Dhariwal, Song and Sutskever 2018), HOList (Bansal, Loos, Rabe, Szegedy and Wilcox 2019). It is still pursued now but usually as part of a system including a language model.

The use of ML in automated proving is still experimental but shows steady progress. One paradigm (almost universal until recently) is reinforcement learning (RL). A prover must make choices along the way: choice of previously proven results to use (premise selection), hypotheses to consider (such as "set $x =$ TRUE" in *SAT*), and choice of rewriting rules and other "tactics" to apply. Many systems follow hand-coded heuristic rules to make these choices.

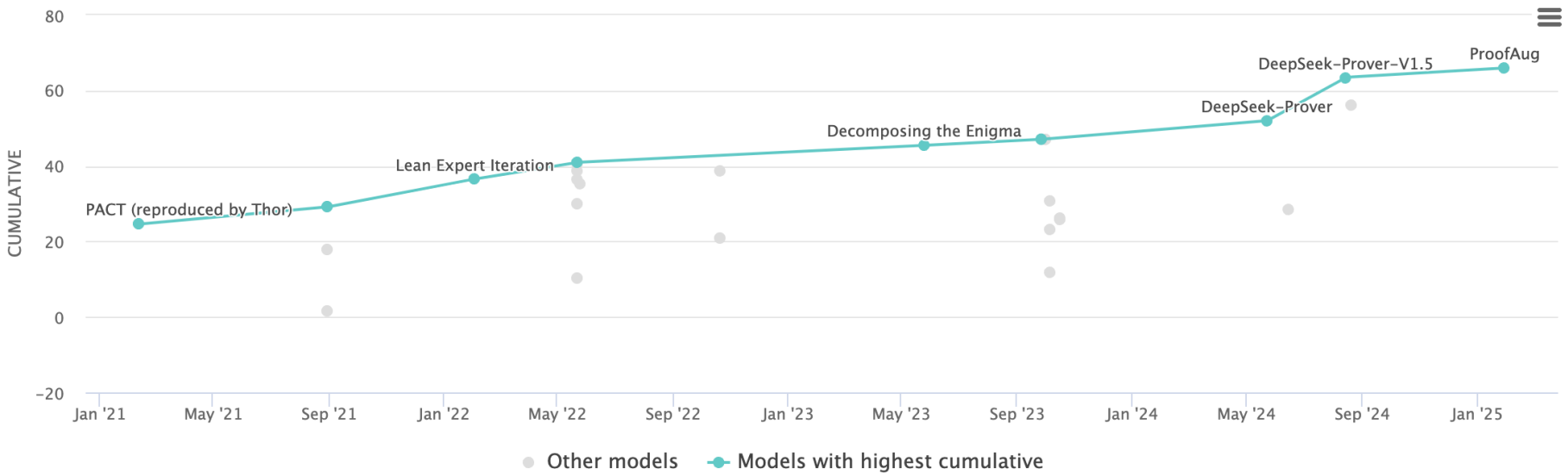
But in RL, the system learns to make these choices, most famously as done by AlphaZero. The choices are actions, like moves in a game of solitaire. A complete proof gets a reward (winning the game).

Early works on HOL proving following this approach include TacticToe (Gauthier, Kalisczyk and Urban 2017), GamePad (Huang, Dhariwal, Song and Sutskever 2018), HOList (Bansal, Loos, Rabe, Szegedy and Wilcox 2019). It is still pursued now but usually as part of a system including a language model.

# An example of autoformalization from the miniF2F benchmark.

Two non-zero real numbers, $a$ and $b$, satisfy $ab = a - b$. Which of the following is a possible value of $\frac{a}{b} + \frac{b}{a} - ab$? (A) -2 (B) $\frac{-1}{2}$ (C) $\frac{1}{3}$ (D) $\frac{1}{2}$ (E) 2

```
theorem amc12_2000_p11
  (a b : ℝ)
  (h₀ : a ≠ 0 ∧ b ≠ 0)
  (h₁ : a * b = a - b) :
  a / b + b / a - a * b = 2 :=
begin
  field_simp [h₀.1, h₀.2],
  simp only [h₁, mul_comm, mul_sub],
  ring,
end
```




From Zheng *et al* 2109.00110 and `paperswithcode.com`

# Large Language Models (LLMs)

This is by far the most significant advance in AI of recent years.
To briefly summarize,

- Statistical language models: predict $P(w_{n+1}|w_1 \ldots w_n)$.
- Model $P(\ldots)$ with a transformer model using attention.
- Train on a large corpus of natural language and code (the "entire internet").
- Generate text completions autoregressively (one word at a time).

Show many surprising abilities requiring "intelligence."

- In-context learning: instead of fine tuning on many examples of a task, give a few examples in the prompt, or just describe it.
- Chain of thought reasoning – fine tune on structured text completions (or in context).
- Reinforcement learning by human feedback – fine tune using another reward signal.

# Reasoning

By reasoning, people mean abilities such as analysis and synthesis (breaking a problem into parts, solving these and putting the solutions together); developing chains of argument; ability to make hypotheses and take them back; and sound deduction, ability to correct mistakes and attribution. Until recently LLMs (say GPT-4) were poor at this.

A popular way to think about how this goes beyond earlier AI (Anthony *et al* 1705.08439, Bengio 2019) is the "system 1/system 2" concept in psychology (Kahneman's terminology, also called dual process theory). System 1 does fast "intuitive" thinking and is related to autoregressive text generation. System 2 which can do reasoning requires conscious effort, and is related to other AI inference methods, such as search.

Another popular concept is the "training time/inference time tradeoff." For example, Jones 2104.03113 studied AlphaZero's ability to learn to play the game of Hex. He found that for an additional $10\times$ training compute, about $15\times$ of test-time compute can be eliminated.

# Reasoning

By reasoning, people mean abilities such as analysis and synthesis (breaking a problem into parts, solving these and putting the solutions together); developing chains of argument; ability to make hypotheses and take them back; and sound deduction, ability to correct mistakes and attribution. Until recently LLMs (say GPT-4) were poor at this.

A popular way to think about how this goes beyond earlier AI (Anthony *et al* 1705.08439, Bengio 2019) is the "system 1/system 2" concept in psychology (Kahneman's terminology, also called dual process theory). System 1 does fast "intuitive" thinking and is related to autoregressive text generation. System 2 which can do reasoning requires conscious effort, and is related to other AI inference methods, such as search.

Another popular concept is the "training time/inference time tradeoff." For example, Jones 2104.03113 studied AlphaZero's ability to learn to play the game of Hex. He found that for an additional 10× training compute, about 15× of test-time compute can be eliminated.

# Reasoning

By reasoning, people mean abilities such as analysis and synthesis (breaking a problem into parts, solving these and putting the solutions together); developing chains of argument; ability to make hypotheses and take them back; and sound deduction, ability to correct mistakes and attribution. Until recently LLMs (say GPT-4) were poor at this.

A popular way to think about how this goes beyond earlier AI (Anthony *et al* 1705.08439, Bengio 2019) is the "system 1/system 2" concept in psychology (Kahneman's terminology, also called dual process theory). System 1 does fast "intuitive" thinking and is related to autoregressive text generation. System 2 which can do reasoning requires conscious effort, and is related to other AI inference methods, such as search.

Another popular concept is the "training time/inference time tradeoff." For example, Jones 2104.03113 studied AlphaZero's ability to learn to play the game of Hex. He found that for an additional $10\times$ training compute, about $15\times$ of test-time compute can be eliminated.

Reasoning is the hot topic at the industrial AI labs and is making significant progress (OpenAI o1 and o3, DeepSeek R1, AlphaProof). Much of this work is proprietary, but DeepSeek has been more (though not entirely) forthcoming about their methods.

A common thread is the use of RL. The validity of solutions to reasoning problems cannot be judged token by token, so one needs to incorporate other reward signals. Ideally these are generated by computer, for example proof verification. At the other extreme one can ask humans to judge validity. One can leverage these judgments by training a reward model to reproduce them – RLHF. A more recent idea is the "process reward model" (Lightman *et al* 2305.20050) which rewards correctly formulated outputs, say chains of thought.

One can use RL or the related supervised fine tuning (labels instead of reward model) to use validity signals for training. A reward model can also be used at inference time to control search and compute time. DeepSeek 2501.12948 claims that R1 does not do extra inference-time compute (unlike DeepSeek-Prover-V1.5 which uses MCTS).

Reasoning is the hot topic at the industrial AI labs and is making significant progress (OpenAI o1 and o3, DeepSeek R1, AlphaProof). Much of this work is proprietary, but DeepSeek has been more (though not entirely) forthcoming about their methods.

A common thread is the use of RL. The validity of solutions to reasoning problems cannot be judged token by token, so one needs to incorporate other reward signals. Ideally these are generated by computer, for example proof verification. At the other extreme one can ask humans to judge validity. One can leverage these judgments by training a reward model to reproduce them – RLHF. A more recent idea is the "process reward model" (Lightman *et al* 2305.20050) which rewards correctly formulated outputs, say chains of thought.

One can use RL or the related supervised fine tuning (labels instead of reward model) to use validity signals for training. A reward model can also be used at inference time to control search and compute time. DeepSeek 2501.12948 claims that R1 does not do extra inference-time compute (unlike DeepSeek-Prover-V1.5 which uses MCTS).

Reasoning is the hot topic at the industrial AI labs and is making significant progress (OpenAI o1 and o3, DeepSeek R1, AlphaProof). Much of this work is proprietary, but DeepSeek has been more (though not entirely) forthcoming about their methods.

A common thread is the use of RL. The validity of solutions to reasoning problems cannot be judged token by token, so one needs to incorporate other reward signals. Ideally these are generated by computer, for example proof verification. At the other extreme one can ask humans to judge validity. One can leverage these judgments by training a reward model to reproduce them – RLHF. A more recent idea is the "process reward model" (Lightman *et al* 2305.20050) which rewards correctly formulated outputs, say chains of thought.

One can use RL or the related supervised fine tuning (labels instead of reward model) to use validity signals for training. A reward model can also be used at inference time to control search and compute time. DeepSeek 2501.12948 claims that R1 does not do extra inference-time compute (unlike DeepSeek-Prover-V1.5 which uses MCTS).

# Reasoning LLMs and Deep Research

I'm currently gathering information on the experimental search for supersymmetry, including past results, upcoming searches in the next five years, and predictions from string theory models. This may take a bit of time to ensure a comprehensive overview. Is there any specific experiment or aspect of string theory you'd like me to prioritize?

discoveries at a future 100 TeV proton antiproton collider

I will gather information on the experimental search for supersymmetry, including results to date, upcoming searches in the next five years, and relevant predictions from string theory models. Additionally, I will look into potential discoveries at a future 100 TeV proton-antiproton collider and how such a machine could contribute to supersymmetry searches. I'll update you once I have a detailed overview.

Query using GPT-4o mini on Feb 4, 2025.

# Prospects at a Future 100 TeV Collider

Given the challenges at the LHC, there is strong interest in a future high-energy hadron collider (perhaps around mid-century) that could dramatically extend the reach for new physics. A **100 TeV proton-proton collider** (such as the proposed FCC-hh at CERN or SppC in China) would be a powerful machine for discovering or definitively ruling out TeV-scale supersymmetry. Here we discuss what such a collider could achieve for SUSY searches and the implications:

- **Dramatically Improved Sensitivity:** A jump from 14 TeV (LHC) to 100 TeV center-of-mass energy would enormously increase the production cross sections for heavy particles. Reactions that are kinematically impossible at the LHC (e.g. producing a pair of 5–10 TeV particles) would become accessible. The parton luminosity for high invariant mass collisions grows rapidly with energy – at 100 TeV, there is substantial quark and gluon flux to produce objects in the ~10–20 TeV mass range INDICO.CERN.CH INDICO.CERN.CH. In addition, a future collider would likely be designed for high luminosity (order 20–30 $ab^{-1}$ over its lifetime). Combined, these factors mean that a 100 TeV collider could search for SUSY particles an order of magnitude heavier than what the LHC can handle. Backgrounds from Standard Model processes, though higher in absolute terms, can be mitigated with advanced detectors and analysis techniques building on LHC experience. In practical terms, where the HL-LHC might have a 5σ discovery reach for gluinos of ~2.5 TeV, a 100 TeV collider could extend that reach by a factor of 5 or more in mass.
- **Mass Reach for Superpartners:** Studies of SUSY potential at a 100 TeV machine show impressive numbers. **Gluinos** (which are among the easiest SUSY particles to produce, via QCD

# Theoretical Predictions from String Theory Models

Supersymmetry has deep roots in theoretical physics, particularly in superstring theory. In superstring models, SUSY is often an essential ingredient to ensure mathematical consistency (e.g. cancelling infinities in vacuum energies) and a quasi-realistic low-energy spectrum. **String theory does not predict a unique scale for supersymmetry breaking**, but it provides a framework in which different SUSY-breaking scenarios can be realized. Over decades, theorists have developed a variety of models for how SUSY might break and what the superpartner spectrum might look like, often guided by string theory or high-scale physics. Below we highlight some key scenarios and their predictions, as well as how string-theoretic considerations (e.g. the "landscape" of vacua) influence expectations:

- **Gravity-Mediated SUSY Breaking (Supergravity models):** In many string-inspired setups, SUSY breaking in a hidden sector is communicated to the visible sector through gravitational interactions. This leads to the **mSUGRA or CMSSM scenario** (constrained MSSM) with a few universal parameters at the grand unification (GUT) scale. These models historically predicted superpartners not far above the electroweak scale. A typical mSUGRA spectrum had squarks and gluinos on the order of a few hundred GeV to 1 TeV, sleptons somewhat lighter, and a light neutralino as LSP – all consistent with solving the hierarchy problem if discovered around the LHC scale ATLAS.CERN. However, the non-observation of SUSY at those scales has forced the parameter space of such models into more fine-tuned regimes: for example, to get a 125 GeV Higgs, **stop squarks** in gravity-mediation must be very heavy (~5–10 TeV) or have large tri-linear couplings (A-terms) ATLAS.CERN. Many gravity-mediated models (including simple string

# AlphaGeometry and AlphaProof

These SOTA systems from DeepMind achieve medal-winning performance on International Mathematical Olympiad problems.

AlphaGeometry (Trinh *et al* Nature 2024) was the subject of a CMSA talk by Trinh last March. It works along the lines we outlined, with a LM trained on synthetic data (randomly generated premises) and a formal prover specialized for geometry. The prover is very good at deduction but cannot introduce new points, lines, *etc.*. This is done by the LM, whose training process implicitly does automated refactoring.

AlphaGeometry2 just appeared (Chervonyi *et al* 2502.03544). It has many technical improvements and "surpasses an average gold medalist," solving 42/50 test problems.

As yet there is no paper documenting AlphaProof. There will be a CMSA talk on it by lead author Thomas Hubert on March 26.

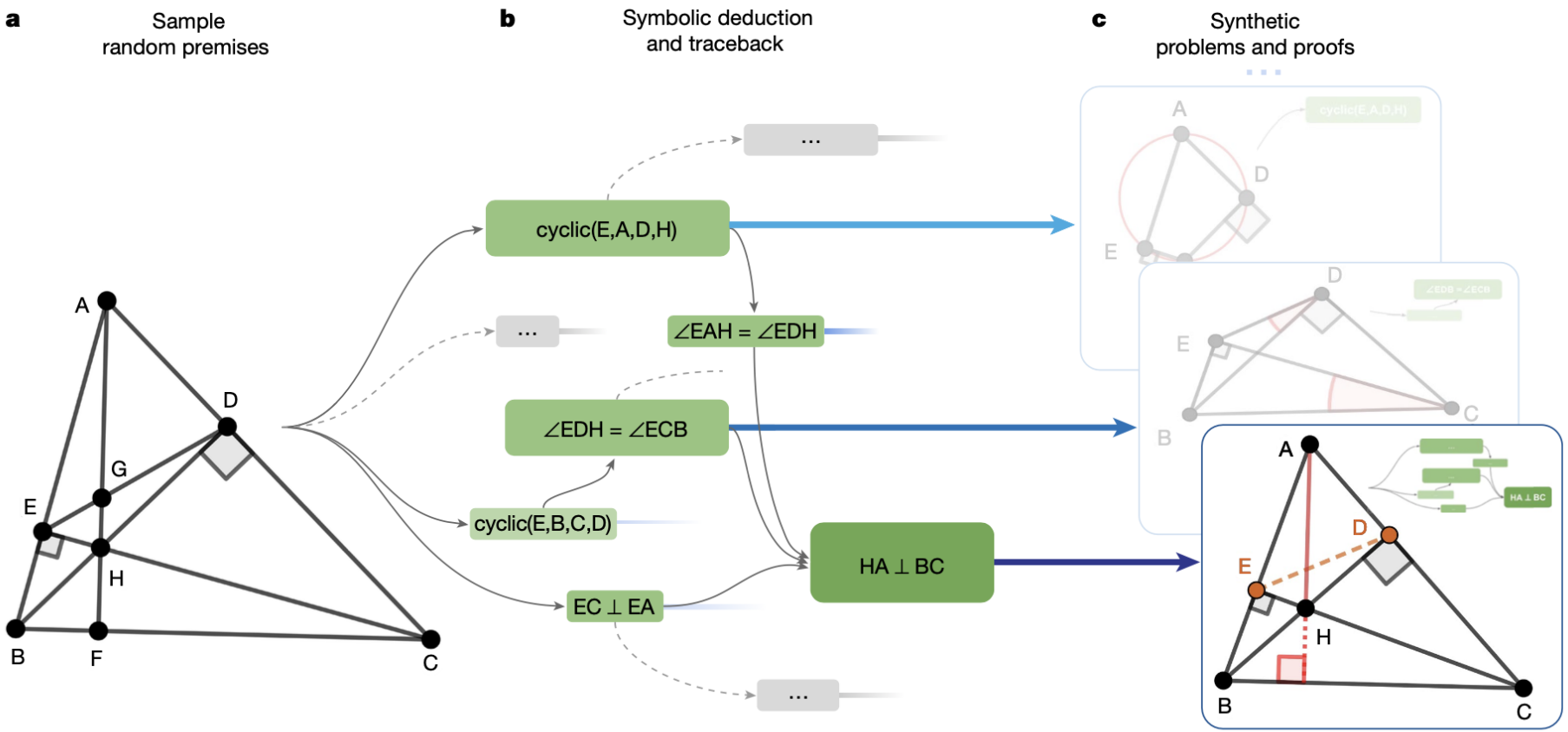
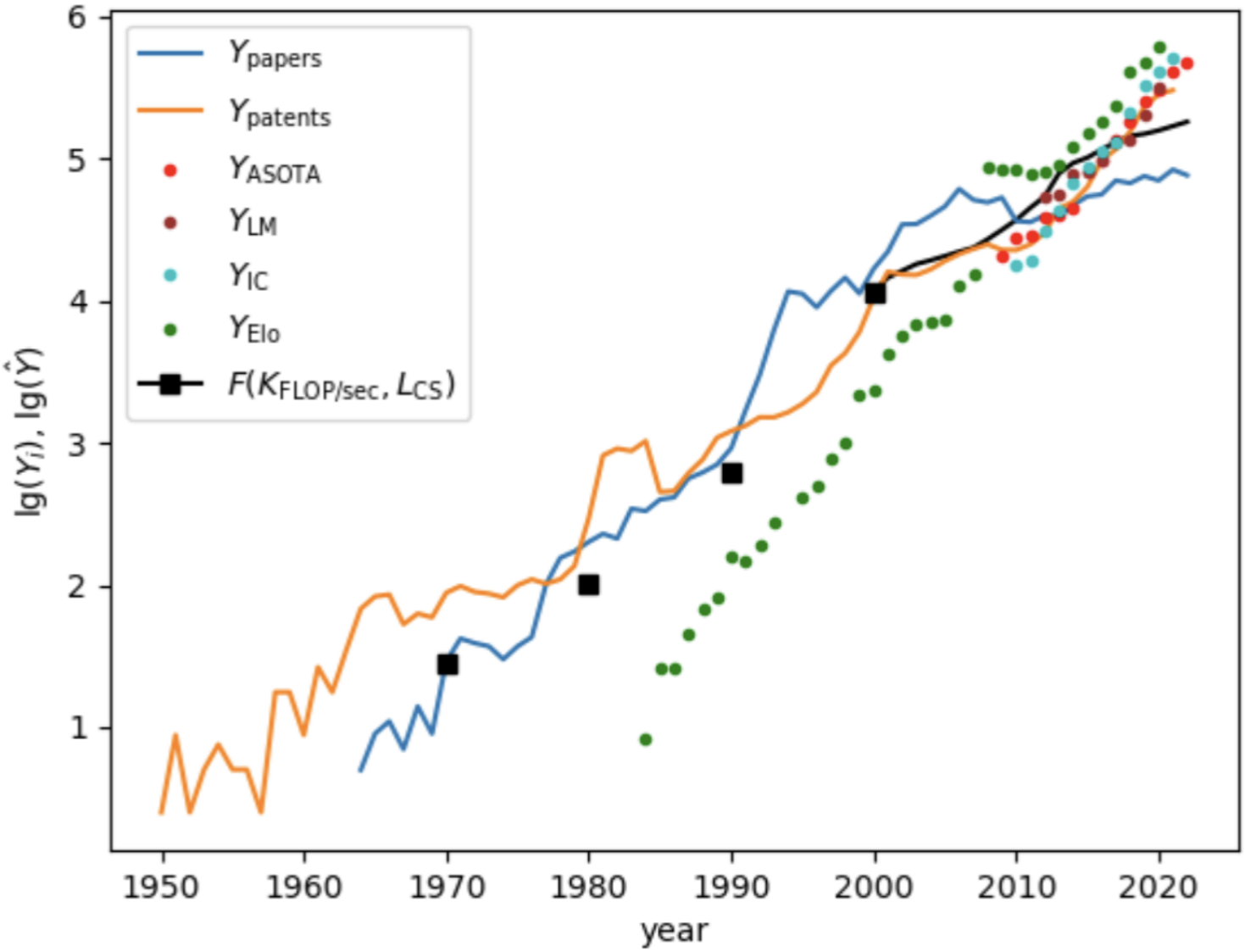



**Fig. 3 | AlphaGeometry synthetic-data-generation process. a**, We first sample a large set of random theorem premises. **b**, We use the symbolic deduction engine to obtain a deduction closure. This returns a directed acyclic graph of statements. For each node in the graph, we perform traceback to find its minimal set of necessary premise and dependency deductions. For example, for the rightmost node 'HA ⊥ BC', traceback returns the green subgraph. **c**, The minimal premise and the corresponding subgraph constitute a synthetic problem and its solution. In the bottom example, points E and D took part in the proof despite being irrelevant to the construction of HA and BC; therefore, they are learned by the language model as auxiliary constructions.

From Trinh *et al*, "Solving olympiad geometry...," Nature 2024.

# We are still riding the exponential



# The computer chess analogy

Computers have become superhuman at a variety of tasks and we can base projections on these past episodes.
Computer chess is an intellectual ability with some (loose) analogies to math/science research, for which this happened in the 1990's.

Chess ranking is relatively objective – in the Elo system, a $2\sigma$ increase in win probability (from 50% to 95%) corresponds to 400 points.

The 2015 FIDE list of players with active rankings:

| | | |
|---|---|---|
| 5,323 players | 2200 to 2299 | Candidate Master |
| 2,869 players | 2300 to 2399 | FIDE Master |
| 1,420 players | 2400 to 2499 | International Master |
| 542 players | 2500 to 2599 | most International Grandmaster |
| 187 players | 2600 to 2699 | all International Grandmaster |
| 40 players | 2700 to 2799 | |
| 4 players | 2800 and above | (Magnus Carlsen was rated 2853) |

# The computer chess analogy

Computers have become superhuman at a variety of tasks and we can base projections on these past episodes.
Computer chess is an intellectual ability with some (loose) analogies to math/science research, for which this happened in the 1990's.

Chess ranking is relatively objective – in the Elo system, a $2\sigma$ increase in win probability (from 50% to 95%) corresponds to 400 points.

The 2015 FIDE list of players with active rankings:

| | | |
|---|---|---|
| 5,323 players | 2200 to 2299 | Candidate Master |
| 2,869 players | 2300 to 2399 | FIDE Master |
| 1,420 players | 2400 to 2499 | International Master |
| 542 players | 2500 to 2599 | most International Grandmaster |
| 187 players | 2600 to 2699 | all International Grandmaster |
| 40 players | 2700 to 2799 | |
| 4 players | 2800 and above | (Magnus Carlsen was rated 2853) |

# The computer chess analogy

Computers have become superhuman at a variety of tasks and we can base projections on these past episodes.
Computer chess is an intellectual ability with some (loose) analogies to math/science research, for which this happened in the 1990's.

Chess ranking is relatively objective – in the Elo system, a $2\sigma$ increase in win probability (from 50% to 95%) corresponds to 400 points.

The 2015 FIDE list of players with active rankings:

| | | |
|---|---|---|
| 5,323 players | 2200 to 2299 | Candidate Master |
| 2,869 players | 2300 to 2399 | FIDE Master |
| 1,420 players | 2400 to 2499 | International Master |
| 542 players | 2500 to 2599 | most International Grandmaster |
| 187 players | 2600 to 2699 | all International Grandmaster |
| 40 players | 2700 to 2799 | |
| 4 players | 2800 and above | (Magnus Carlsen was rated 2853) |

The Swedish Chess Computer Association (SSDF) has done careful rankings of chess engines (running on PCs) since the 80's (in green on the earlier slide). They find a fairly steady average improvement of 54 Elo points per year.

In 1992 computers were competitive with Masters (2200).
In 1997 Deep Blue famously beat Kasparov using offscale computing resources. By 2004 smaller computers reliably beat International Grandmasters (2800).
Computer go is similar: 2006 MCTS breakthrough to 2016 AlphaGo.

How does this compare to progress in AI, measured by benchmarks? In a recent work (arXiv:2501.17894), Sergiy Verstyuk and I developed an index by combining AI benchmarks (in red). This can be compared to Elo scores by the rule 50%$\to$95% $\sim$ 400 points. Very interestingly, we get roughly the same slope as chess ($\times$10 in $\sim$ 8 years).

So, future AI progress could be analogous to the table of chess rankings, moving down a line every 2 years. Analogies between chess and math/science research ability? Candidate master $\sim$ PhD student?

The Swedish Chess Computer Association (SSDF) has done careful rankings of chess engines (running on PCs) since the 80's (in green on the earlier slide). They find a fairly steady average improvement of 54 Elo points per year.
In 1992 computers were competitive with Masters (2200).
In 1997 Deep Blue famously beat Kasparov using offscale computing resources. By 2004 smaller computers reliably beat International Grandmasters (2800).
Computer go is similar: 2006 MCTS breakthrough to 2016 AlphaGo.

How does this compare to progress in AI, measured by benchmarks? In a recent work (arXiv:2501.17894), Sergiy Verstyuk and I developed an index by combining AI benchmarks (in red). This can be compared to Elo scores by the rule 50%$\to$95% $\sim$ 400 points. Very interestingly, we get roughly the same slope as chess ($\times$10 in $\sim$ 8 years).

So, future AI progress could be analogous to the table of chess rankings, moving down a line every 2 years. Analogies between chess and math/science research ability? Candidate master $\sim$ PhD student?

The Swedish Chess Computer Association (SSDF) has done careful rankings of chess engines (running on PCs) since the 80's (in green on the earlier slide). They find a fairly steady average improvement of 54 Elo points per year.
In 1992 computers were competitive with Masters (2200).
In 1997 Deep Blue famously beat Kasparov using offscale computing resources. By 2004 smaller computers reliably beat International Grandmasters (2800).
Computer go is similar: 2006 MCTS breakthrough to 2016 AlphaGo.

How does this compare to progress in AI, measured by benchmarks? In a recent work (arXiv:2501.17894), Sergiy Verstyuk and I developed an index by combining AI benchmarks (in red). This can be compared to Elo scores by the rule 50%$\rightarrow$95% $\sim$ 400 points. Very interestingly, we get roughly the same slope as chess ($\times$10 in $\sim$ 8 years).

So, future AI progress could be analogous to the table of chess rankings, moving down a line every 2 years. Analogies between chess and math/science research ability? Candidate master $\sim$ PhD student?

The Swedish Chess Computer Association (SSDF) has done careful rankings of chess engines (running on PCs) since the 80's (in green on the earlier slide). They find a fairly steady average improvement of 54 Elo points per year.
In 1992 computers were competitive with Masters (2200).
In 1997 Deep Blue famously beat Kasparov using offscale computing resources. By 2004 smaller computers reliably beat International Grandmasters (2800).
Computer go is similar: 2006 MCTS breakthrough to 2016 AlphaGo.

How does this compare to progress in AI, measured by benchmarks?
In a recent work (arXiv:2501.17894), Sergiy Verstyuk and I developed an index by combining AI benchmarks (in red). This can be compared to Elo scores by the rule 50%$\to$95% $\sim$ 400 points. Very interestingly, we get roughly the same slope as chess ($\times$10 in $\sim$ 8 years).

So, future AI progress could be analogous to the table of chess rankings, moving down a line every 2 years. Analogies between chess and math/science research ability? Candidate master $\sim$ PhD student?

The Swedish Chess Computer Association (SSDF) has done careful rankings of chess engines (running on PCs) since the 80's (in green on the earlier slide). They find a fairly steady average improvement of 54 Elo points per year.
In 1992 computers were competitive with Masters (2200).
In 1997 Deep Blue famously beat Kasparov using offscale computing resources. By 2004 smaller computers reliably beat International Grandmasters (2800).
Computer go is similar: 2006 MCTS breakthrough to 2016 AlphaGo.

How does this compare to progress in AI, measured by benchmarks? In a recent work (arXiv:2501.17894), Sergiy Verstyuk and I developed an index by combining AI benchmarks (in red). This can be compared to Elo scores by the rule 50%$\to$95% $\sim$ 400 points. Very interestingly, we get roughly the same slope as chess ($\times$10 in $\sim$ 8 years).

So, future AI progress could be analogous to the table of chess rankings, moving down a line every 2 years. Analogies between chess and math/science research ability? Candidate master $\sim$ PhD student?

The Swedish Chess Computer Association (SSDF) has done careful rankings of chess engines (running on PCs) since the 80's (in green on the earlier slide). They find a fairly steady average improvement of 54 Elo points per year.
In 1992 computers were competitive with Masters (2200).
In 1997 Deep Blue famously beat Kasparov using offscale computing resources. By 2004 smaller computers reliably beat International Grandmasters (2800).
Computer go is similar: 2006 MCTS breakthrough to 2016 AlphaGo.

How does this compare to progress in AI, measured by benchmarks? In a recent work (arXiv:2501.17894), Sergiy Verstyuk and I developed an index by combining AI benchmarks (in red). This can be compared to Elo scores by the rule 50%$\to$95% $\sim$ 400 points. Very interestingly, we get roughly the same slope as chess ($\times$10 in $\sim$ 8 years).

So, future AI progress could be analogous to the table of chess rankings, moving down a line every 2 years. Analogies between chess and math/science research ability? Candidate master $\sim$ PhD student?

# Mathematical research assistants

Let's come back to the wish list from the early slides. The concept of a math/science research assistant or "copilot" is not new – for example in 1987, Gerry Sussman proposed a "Dynamicist's Workbench" to help run and interpret computer experiments in ODE. But it is just now coming in reach, perhaps applied to the successful 2025 paradigms.

One's first conception of this would be a system with which one communicates and collaborates as with a human mathematician. This certainly makes sense but there are more possibilities.

Assisting with coding (autocomplete, search, writing programs from natural language specifications) is one of the most commercially successful applications of LLMs to date.
A direct analog for mathematics is autoformalization and informalization, translating to/from ITP languages.
It seems very possible that even a supersmart AI mathematician will use ITP for the same reasons as humans.

# Mathematical research assistants

Let's come back to the wish list from the early slides. The concept of a math/science research assistant or "copilot" is not new – for example in 1987, Gerry Sussman proposed a "Dynamicist's Workbench" to help run and interpret computer experiments in ODE. But it is just now coming in reach, perhaps applied to the successful 2025 paradigms.

One's first conception of this would be a system with which one communicates and collaborates as with a human mathematician. This certainly makes sense but there are more possibilities.

Assisting with coding (autocomplete, search, writing programs from natural language specifications) is one of the most commercially successful applications of LLMs to date.
A direct analog for mathematics is autoformalization and informalization, translating to/from ITP languages.
It seems very possible that even a supersmart AI mathematician will use ITP for the same reasons as humans.

# Mathematical research assistants

Let's come back to the wish list from the early slides. The concept of a math/science research assistant or "copilot" is not new – for example in 1987, Gerry Sussman proposed a "Dynamicist's Workbench" to help run and interpret computer experiments in ODE. But it is just now coming in reach, perhaps applied to the successful 2025 paradigms.

One's first conception of this would be a system with which one communicates and collaborates as with a human mathematician. This certainly makes sense but there are more possibilities.

Assisting with coding (autocomplete, search, writing programs from natural language specifications) is one of the most commercially successful applications of LLMs to date.
A direct analog for mathematics is autoformalization and informalization, translating to/from ITP languages.
It seems very possible that even a supersmart AI mathematician will use ITP for the same reasons as humans.

ITP is not everything. If one looks at math and physics papers and textbooks, a good fraction of the text does not have a clear translation into formal logic. This includes

- Workflows – sequences of computations and/or proofs which accomplish higher level, more conceptual goals. Example: given an ODE and initial conditions, do an accurate numerical simulation. Even if we use library software, there are many methods to choose from, tests to make on the results, *etc.*.
- Motivations – for intermediate steps in computations and proofs, for definitions, *etc.*. In particular, examples and counterexamples are motivated definitions.
- Difficulty estimates - this problem is easy, that one hard.
- Intuitive connections between different problems, subfields, *etc.*.

While one can discuss all of this in natural language and existing mathematical and programming notations, we may also use new formal languages, as is done in CS and AI.

## So why aren't we using LLMs as math assistants now?

An LLM by itself is not as helpful as a system which can use a broad range of computational methods. We need an LLM which can use tools: not just ITP but also symbolic algebra, and which can code and run programs. This all seems doable but requires work.

There are also questions of reliability. An assistant need not be perfectly reliable, but an error rate $\epsilon$ limits the length of the arguments it can make without help to $\sim 1/\epsilon$. Since mathematics depends on long chains of reasoning, this is a limiting factor. But the error rates, and ability to recognize and correct errors, continue to improve.

OpenAI's Deep Research is the best math/physics AI research tool I have seen so far – try it!
Recent works on AI scientists and co-scientists (see the next slide). Mostly biology/biomedical – different challenges, especially vastness of literature.

So why aren't we using LLMs as math assistants now?

An LLM by itself is not as helpful as a system which can use a broad range of computational methods. We need an LLM which can use tools: not just ITP but also symbolic algebra, and which can code and run programs. This all seems doable but requires work.

There are also questions of reliability. An assistant need not be perfectly reliable, but an error rate $\epsilon$ limits the length of the arguments it can make without help to $\sim 1/\epsilon$. Since mathematics depends on long chains of reasoning, this is a limiting factor. But the error rates, and ability to recognize and correct errors, continue to improve.

OpenAI's Deep Research is the best math/physics AI research tool I have seen so far – try it!
Recent works on AI scientists and co-scientists (see the next slide). Mostly biology/biomedical – different challenges, especially vastness of literature.

So why aren't we using LLMs as math assistants now?

An LLM by itself is not as helpful as a system which can use a broad range of computational methods. We need an LLM which can use tools: not just ITP but also symbolic algebra, and which can code and run programs. This all seems doable but requires work.

There are also questions of reliability. An assistant need not be perfectly reliable, but an error rate $\epsilon$ limits the length of the arguments it can make without help to $\sim 1/\epsilon$. Since mathematics depends on long chains of reasoning, this is a limiting factor. But the error rates, and ability to recognize and correct errors, continue to improve.

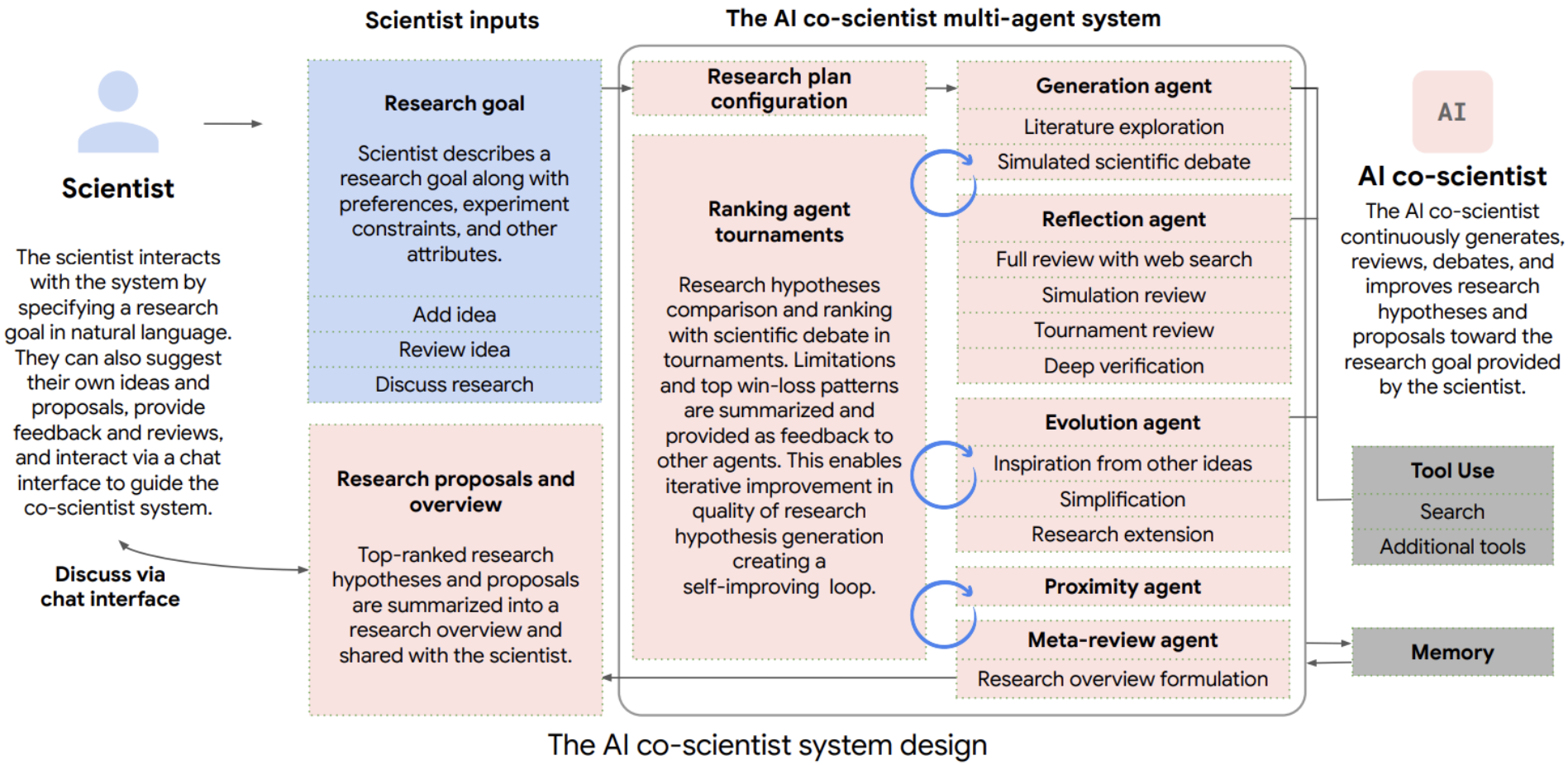
OpenAI's Deep Research is the best math/physics AI research tool I have seen so far – try it!
Recent works on AI scientists and co-scientists (see the next slide).
Mostly biology/biomedical – different challenges, especially vastness of literature.

So why aren't we using LLMs as math assistants now?

An LLM by itself is not as helpful as a system which can use a broad range of computational methods. We need an LLM which can use tools: not just ITP but also symbolic algebra, and which can code and run programs. This all seems doable but requires work.

There are also questions of reliability. An assistant need not be perfectly reliable, but an error rate $\epsilon$ limits the length of the arguments it can make without help to $\sim 1/\epsilon$. Since mathematics depends on long chains of reasoning, this is a limiting factor. But the error rates, and ability to recognize and correct errors, continue to improve.

OpenAI's Deep Research is the best math/physics AI research tool I have seen so far – try it!
Recent works on AI scientists and co-scientists (see the next slide). Mostly biology/biomedical – different challenges, especially vastness of literature.




The AI co-scientist system design

From Gottweis *et al*, “Towards an AI co-scientist,” 2025.

Some near term ideas for AI math assistants and copilots:

- An ITP system which looks like Lean with Visual Studio, but displays the informal and formal texts side-by-side, can autocomplete, do search, translation, *etc.* (Patrick Masson and Kyle Miller)
- The same, adding large databases of examples and counterexamples to help check and generate conjectures. Automated search for examples and counterexamples.
- A "dynamicist's workbench" which takes ODE's (or PDE's), designs and performs numerical experiments, and (helps to) work out the qualitative behavior.

These are not really new ideas, but past implementations bogged down on the cost of programming special purpose systems and the difficulty of working computationally with diverse claims and facts from many sources. The hope is that with LLMs and modern AI, a system can be more general and will not require formalizing everything.

The 2025 math-AI paradigms use structured datasets and ML, require coding to put the pieces together, and present conclusions piecemeal in unstructured research papers. Previous thinking was to systematize the unstructured part – say, in the dynamicist's workbench, develop a formal language for the qualitative behavior of a dynamical system.

LLMs are changing the paradigm to instead work with unstructured data and replace coding with natural language prompts. Continuing on the dynamics example, some systems (say our Solar System) are much studied and there are many results spread among many papers. There are many datasets for comparable observed systems (exoplanets), again spread around. There are many methods, software packages, *etc.*. Rather than bring all of this into one centralized system, a general AI assistant might take a "bibliography" or "dependency file," identify the relevant material, and make it available for standard workflows: static analyses using the Hamiltonian, simulations, complete analyses combining these ingredients.

This will be evolutionary in using the technology of the first (2025) part of the talk, but so much easier that the impact might be revolutionary.

The 2025 math-AI paradigms use structured datasets and ML, require coding to put the pieces together, and present conclusions piecemeal in unstructured research papers. Previous thinking was to systematize the unstructured part – say, in the dynamicist's workbench, develop a formal language for the qualitative behavior of a dynamical system. LLMs are changing the paradigm to instead work with unstructured data and replace coding with natural language prompts. Continuing on the dynamics example, some systems (say our Solar System) are much studied and there are many results spread among many papers. There are many datasets for comparable observed systems (exoplanets), again spread around. There are many methods, software packages, *etc.*. Rather than bring all of this into one centralized system, a general AI assistant might take a "bibliography" or "dependency file," identify the relevant material, and make it available for standard workflows: static analyses using the Hamiltonian, simulations, complete analyses combining these ingredients.
This will be evolutionary in using the technology of the first (2025) part of the talk, but so much easier that the impact might be revolutionary.

# Autonomous mathematical discovery

Despite all the advances in AI in general and theorem proving in particular, it still seems fair to say that, as of today,

*Computers have discovered* **no** *new mathematics by themselves.*

All of the meaningful proofs are of statements given by humans, and all of the discoveries are made by humans using the computers.

And this is not just mathematics which is new to humans. One could ask a computer to emulate historical discoveries. Suppose it is given only the knowledge of human mathematicians circa 1800.
Could it discover group theory and Galois theory?
Could it discover the constructions of the real numbers?
Even simpler, suppose it knows about the integers, could it discover a correct definition of the rationals? And realize that it has made a "discovery" ?

# Autonomous mathematical discovery

Despite all the advances in AI in general and theorem proving in particular, it still seems fair to say that, as of today,

*Computers have discovered* **no** *new mathematics by themselves.*

All of the meaningful proofs are of statements given by humans, and all of the discoveries are made by humans using the computers.

And this is not just mathematics which is new to humans. One could ask a computer to emulate historical discoveries. Suppose it is given only the knowledge of human mathematicians circa 1800.
Could it discover group theory and Galois theory?
Could it discover the constructions of the real numbers?
Even simpler, suppose it knows about the integers, could it discover a correct definition of the rationals? And realize that it has made a "discovery" ?

# Autonomous mathematical discovery

Despite all the advances in AI in general and theorem proving in particular, it still seems fair to say that, as of today,

*Computers have discovered* **no** *new mathematics by themselves.*

All of the meaningful proofs are of statements given by humans, and all of the discoveries are made by humans using the computers.

And this is not just mathematics which is new to humans. One could ask a computer to emulate historical discoveries. Suppose it is given only the knowledge of human mathematicians circa 1800.
Could it discover group theory and Galois theory?
Could it discover the constructions of the real numbers?
Even simpler, suppose it knows about the integers, could it discover a correct definition of the rationals? And realize that it has made a "discovery" ?

This challenge of autonomous mathematical discovery has not received its due. It is hard to believe a computer will (prove the Riemann hypothesis, or fill in the blank yourself) when it has no ability to make its own discoveries. A pioneer in AMD is David McAllester with his "MathZero" proposal (2005.05512).

- There is an algorithm to generate all provable statements (brute force enumeration of proofs). In a sense the challenge is to select the interesting statements from these, and justify the selection.
- A strain of AI work in the 90's attempted this, as reviewed in a 2000 survey of Colton, Bundy and Walsh. On March 12 (tomorrow), Randy Davila will speak at CMSA on his system TxGraffiti.
- This work is proof of concept but still needs a human user and hand coding, so doesn't readily scale.
- The Mimino system of Poesia *et al* 2407.00695 discovers simple lemmas in propositional logic, arithmetic and group theory.
- One can propose objective functions for "interestingness," and multiagent systems which might evolve them – see my April 2024 Univ of Cambridge talk.

This challenge of autonomous mathematical discovery has not received its due. It is hard to believe a computer will (prove the Riemann hypothesis, or fill in the blank yourself) when it has no ability to make its own discoveries. A pioneer in AMD is David McAllester with his "MathZero" proposal (2005.05512).

- There is an algorithm to generate all provable statements (brute force enumeration of proofs). In a sense the challenge is to select the interesting statements from these, and justify the selection.
- A strain of AI work in the 90's attempted this, as reviewed in a 2000 survey of Colton, Bundy and Walsh. On March 12 (tomorrow), Randy Davila will speak at CMSA on his system TxGraffiti.
- This work is proof of concept but still needs a human user and hand coding, so doesn't readily scale.
- The Mimino system of Poesia *et al* 2407.00695 discovers simple lemmas in propositional logic, arithmetic and group theory.
- One can propose objective functions for "interestingness," and multiagent systems which might evolve them – see my April 2024 Univ of Cambridge talk.

This challenge of autonomous mathematical discovery has not received its due. It is hard to believe a computer will (prove the Riemann hypothesis, or fill in the blank yourself) when it has no ability to make its own discoveries. A pioneer in AMD is David McAllester with his "MathZero" proposal (2005.05512).

- There is an algorithm to generate all provable statements (brute force enumeration of proofs). In a sense the challenge is to select the interesting statements from these, and justify the selection.
- A strain of AI work in the 90's attempted this, as reviewed in a 2000 survey of Colton, Bundy and Walsh. On March 12 (tomorrow), Randy Davila will speak at CMSA on his system TxGraffiti.
- This work is proof of concept but still needs a human user and hand coding, so doesn't readily scale.
- The Mimino system of Poesia *et al* 2407.00695 discovers simple lemmas in propositional logic, arithmetic and group theory.
- One can propose objective functions for "interestingness," and multiagent systems which might evolve them – see my April 2024 Univ of Cambridge talk.

This challenge of autonomous mathematical discovery has not received its due. It is hard to believe a computer will (prove the Riemann hypothesis, or fill in the blank yourself) when it has no ability to make its own discoveries. A pioneer in AMD is David McAllester with his "MathZero" proposal (2005.05512).

- There is an algorithm to generate all provable statements (brute force enumeration of proofs). In a sense the challenge is to select the interesting statements from these, and justify the selection.
- A strain of AI work in the 90's attempted this, as reviewed in a 2000 survey of Colton, Bundy and Walsh. On March 12 (tomorrow), Randy Davila will speak at CMSA on his system TxGraffiti.
- This work is proof of concept but still needs a human user and hand coding, so doesn't readily scale.
- The Mimino system of Poesia *et al* 2407.00695 discovers simple lemmas in propositional logic, arithmetic and group theory.
- One can propose objective functions for "interestingness," and multiagent systems which might evolve them – see my April 2024 Univ of Cambridge talk.

This challenge of autonomous mathematical discovery has not received its due. It is hard to believe a computer will (prove the Riemann hypothesis, or fill in the blank yourself) when it has no ability to make its own discoveries. A pioneer in AMD is David McAllester with his "MathZero" proposal (2005.05512).

- There is an algorithm to generate all provable statements (brute force enumeration of proofs). In a sense the challenge is to select the interesting statements from these, and justify the selection.
- A strain of AI work in the 90's attempted this, as reviewed in a 2000 survey of Colton, Bundy and Walsh. On March 12 (tomorrow), Randy Davila will speak at CMSA on his system TxGraffiti.
- This work is proof of concept but still needs a human user and hand coding, so doesn't readily scale.
- The Mimino system of Poesia *et al* 2407.00695 discovers simple lemmas in propositional logic, arithmetic and group theory.
- One can propose objective functions for "interestingness," and multiagent systems which might evolve them – see my April 2024 Univ of Cambridge talk.

This challenge of autonomous mathematical discovery has not received its due. It is hard to believe a computer will (prove the Riemann hypothesis, or fill in the blank yourself) when it has no ability to make its own discoveries. A pioneer in AMD is David McAllester with his "MathZero" proposal (2005.05512).

- There is an algorithm to generate all provable statements (brute force enumeration of proofs). In a sense the challenge is to select the interesting statements from these, and justify the selection.
- A strain of AI work in the 90's attempted this, as reviewed in a 2000 survey of Colton, Bundy and Walsh. On March 12 (tomorrow), Randy Davila will speak at CMSA on his system TxGraffiti.
- This work is proof of concept but still needs a human user and hand coding, so doesn't readily scale.
- The Mimino system of Poesia *et al* 2407.00695 discovers simple lemmas in propositional logic, arithmetic and group theory.
- One can propose objective functions for "interestingness," and multiagent systems which might evolve them – see my April 2024 Univ of Cambridge talk.

Ultimately the test of true AMD is that the system can build on its own discoveries and continue indefinitely, producing new and interesting statements (under some definition).

Unlike computer chess, here we are suggesting that computers will go from near-zero ability in 2025 to significant (possibly superhuman?) ability in 2035. While this may seem implausible, LLMs have made such leaps of progress for all sorts of unusual and “creative” abilities, such as writing poetry about the infinitude of prime numbers.

It seems plausible that significant mathematical discovery – the definition of truly novel mathematical objects, unexpected connections between distant subfields – requires understanding the structures involved much more deeply than is needed for problem solving or explication. There is no argument forbidding an AI system from achieving this. But training on a dataset of problems and answers, or on the workflows and motivations discussed earlier, may not be enough to stimulate it. In this sense I think more is needed than is present in LLM work so far, but maybe not much more.

Ultimately the test of true AMD is that the system can build on its own discoveries and continue indefinitely, producing new and interesting statements (under some definition).

Unlike computer chess, here we are suggesting that computers will go from near-zero ability in 2025 to significant (possibly superhuman?) ability in 2035. While this may seem implausible, LLMs have made such leaps of progress for all sorts of unusual and “creative” abilities, such as writing poetry about the infinitude of prime numbers.

It seems plausible that significant mathematical discovery – the definition of truly novel mathematical objects, unexpected connections between distant subfields – requires understanding the structures involved much more deeply than is needed for problem solving or explication. There is no argument forbidding an AI system from achieving this. But training on a dataset of problems and answers, or on the workflows and motivations discussed earlier, may not be enough to stimulate it. In this sense I think more is needed than is present in LLM work so far, but maybe not much more.

Ultimately the test of true AMD is that the system can build on its own discoveries and continue indefinitely, producing new and interesting statements (under some definition).

Unlike computer chess, here we are suggesting that computers will go from near-zero ability in 2025 to significant (possibly superhuman?) ability in 2035. While this may seem implausible, LLMs have made such leaps of progress for all sorts of unusual and "creative" abilities, such as writing poetry about the infinitude of prime numbers.

It seems plausible that significant mathematical discovery – the definition of truly novel mathematical objects, unexpected connections between distant subfields – requires understanding the structures involved much more deeply than is needed for problem solving or explication. There is no argument forbidding an AI system from achieving this. But training on a dataset of problems and answers, or on the workflows and motivations discussed earlier, may not be enough to stimulate it. In this sense I think more is needed than is present in LLM work so far, but maybe not much more.

# Conclusions

- Machine learning and other CS technologies such as combinatorial optimization are useful for math/physics research, but as of 2025 have largely led to evolutionary progress (in contrast to the revolutionary progress of AlphaFold).
- Interactive theorem provers such as Lean could help us do mathematics and would bring rigorous proof within the reach of many more scientists (theoretical physicists in particular). While the foundations have been laid, users still face a steep learning curve, and in 2025 the technology is nowhere near as popular as (say) computer algebra.
- Any extrapolation from this baseline must recognize that progress has been exponential so far and there are good reasons to expect this to continue. Worries have been raised (breakdown of Moore's Law, running out of training data, large energy requirements) and clearly will be limiting someday, but not in the next few years.

# Conclusions

- While AI systems are not yet very good math assistants, it seems very possible that they will become so in the short term (by 2026 or 2027). This would fulfill the prediction that "AGI is imminent."
- If so, then on our present exponential trajectory, and if we follow the computer chess analogy, we can expect AIs with strong math/theoretical science research abilities 5 years after that, and abilities matching the all-time greatest human mathematicians after another 5 years.
- This is hard to believe, but there it is.
- Such abilities should open mathematical/scientific vistas which seem ridiculously far beyond us now: deeper mathematics, digital cells, a deep mathematical understanding of evolution modeled on theory of learning, a real understanding of the string landscape. There will be great opportunities and great dangers.