Mathematics in the Age of Large Language Models

Przemyslaw Chojecki

ulam.ai

December 8, 2025

Abstract

Mathematical proofs were long considered the hardest frontier for artificial intelligence. In 2016 the *DeepAlgebra* program proposed an ambitious agenda: to translate informal mathematics, written in LaTeX, into the formal languages of proof assistants such as Coq and Lean, and then to use automated reasoning to explore large theories like the Stacks Project. [1] At the time, this vision relied on speculative deep learning tools and labor-intensive formalization. Since then, large language models (LLMs) and modern proof assistants have transformed the landscape. In 2024–2025, systems such as AlphaProof and AlphaGeometry reached silver-medal performance on International Mathematical Olympiad (IMO) problems, while newer models like DeepSeekMath-V2 and Aristotle achieved gold-level scores on competition benchmarks and even solved a 30-year-old Erdős problem variant in fully formal Lean. [3, 5, 7, 16] In this article I revisit the original DeepAlgebra proposal in light of these developments, survey emerging methods for LLM-assisted theorem proving and autoformalization, and outline a 2025 "DeepAlgebra loop" in which neural search, LLM-based translation and formal verification form a closed neuro-symbolic discovery pipeline. I argue that the most promising path couples LLMs tightly with symbolic verifiers, expands large formal libraries, and designs workflows in which humans and AI systems collaborate on conjecture, proof and exposition. Rather than focusing on isolated leaderboards, progress should be understood as the gradual integration of mathematics into a shared formal ecosystem navigable by both humans and machines.

Introduction

When the *DeepAlgebra* program was first sketched in 2016, the idea that a machine could read research-level LaTeX, translate it into Lean or Coq, and then prove new theorems in algebraic geometry still felt remote. [1] The largest formal projects required years of expert effort to encode a single major theorem. Automated theorem provers were powerful but brittle, and machine learning for proofs was in its infancy.

Two revolutions have unfolded since then. Proof assistants based on dependent type theory, especially Lean, have matured into platforms with large, collaboratively built libraries of formalized mathematics, such as the Lean mathematical library `mathlib`. [2] At the same time, large language models trained on internet-scale text, code and scientific articles have acquired surprising fluency in both informal mathematical language and formal proof scripts.

Together, these developments have brought the DeepAlgebra vision much closer to reality. In the mid-2020s, AlphaProof and AlphaGeometry reached silver-medal performance on IMO-style problems by discovering fully formal Lean proofs, [3, 5] while DeepSeekMath-V2 pushed self-verifiable theorem proving to near-perfect scores on the Putnam 2024 exam and gold-level performance on IMO 2025. [7] In parallel, Harmonic's Aristotle model recently produced a Lean-verified solution to an Erdős problem that had been open for nearly three decades, in roughly six hours of autonomous search. [16, 18]

These milestones suggest a shift: from viewing AI as a toy assistant for textbook problems to treating it as a serious co-pilot for front-line mathematics. Yet many questions remain about scope, reliability, understanding and credit. This Perspective revisits DeepAlgebra in light of these changes. I first recall the original program and its central bottleneck: the cost of formalization. I then summarise the current landscape of LLM-based theorem proving and autoformalization, introduce a unified "DeepAlgebra loop" architecture for neuro-symbolic search, and discuss what AI still cannot do. Finally, I sketch a roadmap towards AI-native mathematics and reflect on how these tools may reshape our understanding of proof and discovery. Similar questions are raised in a recent comment by Naskrecki and Ono, which maps pathways from informal mathematics to machine-augmented discovery. [20]

Key insights in 2025

- **Rapid progress in formal proofs.** Since DeepAlgebra was proposed, LLM-based systems have progressed from solving toy problems to achieving silver- and gold-level performance on IMO-style benchmarks, with near-perfect scores on Putnam 2024 and a fully automated solution to an Erdős problem in Lean. [3, 5, 7, 16]
- **Current methods are effective but limited.** Models like AlphaProof, DeepSeekMath-V2 and Ax-Prover combine reinforcement learning, synthetic training

data and self-verification, and outperform typical humans on formalized competition benchmarks. [3, 7, 6, 9] Nonetheless, they rely heavily on curated problem distributions, remain brittle on open conjectures and require substantial human oversight.

- **Future developments emphasise amplified mathematics.** Large initiatives such as DARPA's Exponentiating Mathematics (expMath) program and Google DeepMind's AI for Math Initiative aim to build AI "co-authors" that can help decompose and prove research-level theorems, linking formal libraries like `mathlib` to databases such as the L-functions and Modular Forms Database (LMFDB). [12, 5, 13]

From LaTeX to Lean: the DeepAlgebra vision

The starting point of DeepAlgebra was pragmatic. Formalizing mathematics is expensive: major projects such as the formal proof of the Feit–Thompson theorem or the Kepler conjecture demanded tens of thousands of lines of code and years of dedicated work. Only a tiny fraction of modern mathematics existed in machine-checkable form.

DeepAlgebra identified a central bottleneck: *autoformalization*, the process of translating human-written proofs into a formal language. To scale beyond isolated projects, three ingredients were proposed: [1]

1. A **dictionary** from LaTeX to a proof assistant's language, capable of recognising definitions, statements and basic proof structure.
2. A way to **connect** this pipeline to automated theorem provers and "hammers", so that once a body of mathematics was encoded, machines could begin to prove new theorems within it.
3. A **testbed** of rich, structured mathematics, such as the Stacks Project, where these techniques could be deployed on a realistic scale.

The proposal anticipated that deep learning would assist with disambiguating notation, clustering related concepts and translating routine proof steps into tactic scripts. [1] In practice, however, the neural components of DeepAlgebra were primarily used as powerful pattern recognisers: they could rank likely premises, suggest which previously known facts might be relevant, and help guess how a proof might continue, but they did not themselves produce rigorous, machine-checkable proofs.

Viewed from 2025, this reveals what I now call the **translation gap**. Neural networks could form rich internal representations of algebraic structure and implicitly "sense" that certain statements were likely to be true, but they lacked any direct way to express this as a formal derivation in Lean or Coq. Turning statistical judgements into symbolic mathematics still required substantial human work.

The core question of DeepAlgebra, in retrospect, was therefore not only how to search mathematical landscapes, but how to *translate* between continuous neural intuition and discrete symbolic proof.

The LLM turn: language models as proof agents

Large language models have changed how we think about reasoning in AI. Models trained on mixtures of natural language, code and mathematical content can already solve many competition-style problems, generate LaTeX solutions, and outline plausible arguments. However, their reasoning is fundamentally statistical rather than deductive: they can produce elegant but incorrect proofs, and their internal justification is opaque.

The natural remedy is to pair LLMs with *formal verification*. A proof assistant becomes an uncompromising referee: the model proposes a proof or tactic, the assistant checks it, and any failure yields structured error messages. This feedback loop underlies several prominent systems:

- **AlphaProof**, which couples a language model with an AlphaZero-style reinforcement learning algorithm in the Lean environment, training on millions of autoformalized problems and reaching silver-medal performance on formalized IMO 2024 tasks. [3, 4]
- **DeepSeek-Prover** and its successors, which fine-tune LLMs on millions of synthetic Lean 4 proofs, achieving state-of-the-art performance on the miniF2F benchmark and solving several formalized IMO problems that earlier models could not. [6]
- **DeepSeekMath-V2**, which explicitly incorporates self-verification—scaling the compute used for checking candidate proofs—to reach gold-level scores on IMO 2025 and a near-perfect 118/120 on Putnam 2024. [7]
- **Ax-Prover**, a multi-agent framework that equips general-purpose LLMs with Lean tools and achieves up to 96% accuracy on a new QuantumTheorems benchmark, while remaining competitive on Putnam-style problems. [9, 10]

In all of these, the role envisaged by DeepAlgebra—AI as a bridge between informal text, heuristic search and formal proof engines—is now played by LLMs fine-tuned for mathematical reasoning. The model is no longer an oracle but a *proposal generator* embedded in a symbolic environment.

Autoformalization: machines learning to read proofs

Perhaps the clearest realisation of the DeepAlgebra agenda is **autoformalization**: turning informal mathematical text into formal statements and, when possible, proofs.

Early experiments focused on constrained settings. Textbook-style statements were mapped to formal goals; competition problems were translated into Lean or Coq; simple proof steps were reconstructed as tactic sequences. More recently, the field has moved towards end-to-end pipelines:

- Benchmarks such as ProofNet [24] that pair informal statements with human-verified formalizations, particularly in Lean, provide training data for models that learn to map natural language to formal syntax. [6, 15]

- LLM-driven systems retrieve relevant lemmas from large formal libraries such as `mathlib` and iteratively refine candidate proofs using error messages from the proof assistant. [6, 14]

- New methods like **StepProof** perform sentence-level verification of natural language proofs, breaking them into smaller claims that are individually checked and, when possible, autoformalized. [11]

These systems already handle significant fragments of undergraduate and some graduate-level mathematics. They struggle with the long, non-linear, heavily cross-referenced structure of research articles, but they demonstrate that a practical DeepAlgebra dictionary is within reach: an AI that reads LaTeX, infers types and scopes of variables, identifies references to existing lemmas, and produces a formal counterpart in a proof assistant.

The vision of machine-readable research monographs—from the Stacks Project to textbooks in functional analysis or representation theory—is no longer purely speculative.

Discovery: AI as a generator of theorems and structures

DeepAlgebra focused primarily on formalizing existing mathematics. Recent work suggests that AI can also assist in *discovering* new results.

One line of research uses self-play and synthetic data. Systems generate random configurations (for example, in Euclidean geometry or algebraic structures), conjecture relations, and attempt to prove them in a formal environment. In Euclidean geometry, for instance, **AlphaGeometry** achieves Olympiad-level performance using self-play and synthetic proof generation. [23] The resulting synthetic theorems and proofs form large corpora that train better proof models, which can then tackle human-written problems at Olympiad level. [3, 4]

Another approach couples LLMs with external evaluators that rigorously score candidate solutions. A language model proposes programs or constructions; an independent checker evaluates them against a precise objective. While this paradigm was first explored in algorithmic domains, its spirit now appears in theorem proving as well: models propose many candidate proofs, of which only those passing formal verification are kept and used for further training. [7, 6]

Within Lean, methods such as **LeanNavigator** explore state-transition graphs of existing proofs to generate millions of new theorems and alternative proofs, greatly expanding the available training data for theorem provers. [8] Other systems, such as LeanConjecturer, explicitly generate new conjectures that can then be attacked by provers.

The most striking example to date of AI-assisted discovery is perhaps Harmonic's **Aristotle** system, which recently produced a Lean-verified solution to an Erdős problem that had been open for nearly 30 years. [16, 17, 18] The system reportedly worked autonomously for around six hours to find a proof, which was then rapidly checked by the Lean kernel.

These developments hint at an expanded DeepAlgebra program: not just encoding human mathematics for machines, but enabling machines to propose new, formally verified mathematics for humans.

The DeepAlgebra loop: neuro-symbolic search in 2025

The original DeepAlgebra work emphasised two ends of a pipeline: neural networks exploring algebraic structures, and interactive theorem provers checking symbolic proofs. In 2025, LLMs allow us to connect these ends into a closed loop that I will call the **DeepAlgebra loop** or **neuro-symbolic search**:

Stage 1: Neural search (the intuition engine)

Continuous optimisation methods—graph neural networks, differentiable loss functions measuring violations of algebraic laws, or learned proposal distributions over algebraic data structures—explore the landscape of candidate structures and identities. The output is a *conjecture object*: a candidate identity, algebraic law or structural pattern that appears to hold in regions of the search space.

Concrete examples include searching for identities in non-associative magmas or for relations among elements in noncommutative rings. The neural engine finds low-loss regions where certain equations appear robust across samples.

Stage 2: Autoformalization (the translator)

An LLM fine-tuned on formal mathematics acts as the interface layer. It receives the conjecture object—for instance, a multiplication table, a set of relations, or a family of examples—together with contextual information from `mathlib` or other libraries. It then proposes:

- a Lean definition capturing the structure suggested by the neural search, and

- a formal theorem statement expressing the conjectured identity or property.

In many cases the LLM can also draft a tactic script or proof outline. When it cannot, it at least produces a precisely stated goal that is now visible to symbolic tools and humans.

Stage 3: Symbolic verification (the judge)

A proof assistant such as Lean 4 acts as the judge. It runs the proof script, checks each step, and either accepts the result (adding a new theorem to the library) or produces structured error messages. These errors feed back into:

- the LLM, which can refine the formalization or attempt alternative proof strategies (including self-verification loops of the kind used in DeepSeekMath-V2), and
- the neural search module, which can adjust its notion of "low-loss" regions in light of which conjectures survived formal scrutiny.

Over time, this loop can in principle become self-improving: formal libraries grow; LLMs are fine-tuned on richer autoformalized corpora; neural search modules learn better priors about which regions of the algebraic landscape are likely to yield formally provable theorems.

A small "toy" experiment already illustrates the idea. One can take a finite algebraic structure discovered by continuous optimisation, present its operation table to an LLM with instructions to define it in Lean and check associativity, and obtain a fully verified formal proof if the structure genuinely satisfies the law. While simple, such examples concretely demonstrate how the translation gap of 2016 is being closed.

Related neuro-symbolic loops have been explored in algorithmic discovery, for example in AlphaTensor for matrix multiplication and FunSearch for combinatorial constructions. [21, 22]

Box 1: The Neuro-Symbolic Discovery Cycle

The **DeepAlgebra loop** is a three-stage neuro-symbolic discovery cycle:

1. **Neural search (intuition).** Continuous optimisation or GNN-based modules explore spaces of algebraic structures, minimising losses that measure violations of axioms and producing *conjecture objects* such as candidate identities or operation tables.
2. **Autoformalization (translation).** An LLM trained on formal mathematics takes a conjecture object together with context from libraries like `mathlib` and emits a Lean 4 definition, a precise theorem statement, and often a draft proof script.
3. **Symbolic verification (judgement and feedback).** The Lean kernel checks the script. On success, the theorem is added to the library; on failure, structured error messages are fed back to the LLM and, when appropriate, to the neural search module, refining both translation and search.

In this way, speculative neural "hallucinations" become useful: the system may generate many risky candidates, but only those accepted by the proof kernel enter the mathematical record.

Implications for mathematical practice

Trustless but not trust-free

The neuro-symbolic nature of modern AI mathematics suggests a form of "trustless" collaboration: every lemma is checked by a small, auditable kernel, independent of the heuristics used to find it. Projects such as StepProof push this further by decomposing natural-language arguments into sentence-level claims that can be individually verified. [11]

However, trust does not disappear; it moves. Mathematicians must decide which libraries and axioms to accept, which proof assistants and kernels to rely on, and which AI pipelines to integrate into their workflows. They must also decide epistemic questions: Is a theorem truly understood if no human has read the proof, even if it is formally correct? How should we weigh a short human proof against a very long AI-generated one?

The Epistemic Shift: Verification vs. Understanding

The neuro-symbolic era forces a decoupling of *verification* from *understanding*. In the DeepAlgebra loop, validity is delegated to the kernel (Lean): assuming the kernel and its hardware/software stack are sound, a checked proof is correct regardless of how it was found. This creates a new category of truth: theorems that are known to be true (verified) but remain opaque to human intuition (unexplained). The role of the mathematician shifts from "certifier of correctness" to "architect of meaning" - interpreting the output of the neuro-symbolic loop to extract conceptual insights from machine-generated proofs.

New research infrastructures

Large-scale initiatives are now explicitly targeting AI-assisted mathematics. DARPA's **Exponentiating Mathematics (expMath)** program aims to develop AI co-authors capable of proposing and proving useful abstractions in pure mathematics. [12] Google DeepMind's **AI for Math Initiative** partners with leading institutions to apply systems such as AlphaProof and Gemini-based models to cutting-edge problems. [5]

Meanwhile, philanthropic efforts such as the **AI for Math Fund** support projects that link databases like LMFDB to formal libraries such as `mathlib`, enabling theorem provers to target specific entries for verification rather than formalizing entire corpora. [13, 14] These efforts indicate a shift from one-off heroic formalizations to a sustained, ecosystem-level approach to AI and mathematics.

Human–AI workflows

In practice, many mathematicians now experiment with workflows along the following lines:

- sketching conjectures and heuristic arguments in informal prose,
- using LLMs to generate examples, counterexamples or algebraic manipulations,
- invoking autoformalization tools or assistants such as Lean-based copilots to produce draft formal proofs, and
- using formal verification to check and refactor arguments.

Case studies already show that such workflows can accelerate debugging and exploration. In anecdotal reports, researchers have used AI tools to rapidly search parameter spaces for counterexamples, to suggest alternative formulations of lemmas, and to formalize results in number theory and cryptography that would otherwise have taken much longer. [9, 18, 13]

What AI still cannot do

Despite these advances, current AI systems fall far short of acting like working research mathematicians.

First, most successes occur in **narrow, well-specified domains**: competition problems, specific areas with dense formalization, or synthetic environments designed around a fixed set of primitives. General-purpose LLMs can sketch plausible arguments but remain brittle on open-ended research questions. Even domain-tuned models routinely make subtle errors that are hard to detect without formal checking. [3, 7, 11]

Second, AI systems excel at *local* reasoning—filling gaps, applying lemmas, exploring variants—but struggle with **global structure**. Tasks such as inventing useful definitions, deciding which conjectures are interesting, reorganising a theory, or choosing the "right" proof among many formally correct options are only weakly captured by existing benchmarks.

Third, there are unresolved issues of **authorship and credit**. Formal verification ensures that each inference step is correct relative to axioms, but does not settle questions such as:

- How should authorship be attributed when an AI system, trained on community-built libraries, produces a decisive lemma or proof?
- What standards should journals and referees adopt for machine-assisted proofs?
- How should we evaluate conceptual understanding versus brute-force search?

These questions are no longer hypothetical. The Aristotle–Erdős result, for example, has already triggered debate about what counts as an acceptable mathematical contribution from an AI system. [17, 19]

Outlook: amplified and AI-native mathematics

Looking ahead, it is tempting to speculate about milestones such as AI-assisted advances on famous conjectures, but it is more important to understand likely qualitative shifts.

In the short term, AI is likely to function as an accelerator and safety net: suggesting lemmas, finding bugs in arguments, and lowering the barrier to entry for formalization. Tools such as StepProof, LeanNavigator and Ax-Prover will continue to improve both the scale and the granularity at which we can verify and search for proofs. [11, 8, 9]

By 2030, if current trends continue, we may see:

- substantial fractions of new results in certain journals accompanied by formal proofs,
- routine use of AI co-authors in large collaborative projects, as envisaged by expMath and related initiatives, and
- richer forms of meta-mathematical analysis, where networks of formal proofs are mined for patterns, analogies and hidden structure.

At that point, the distinction between "computer-assisted" and "AI-driven" mathematics may blur. DeepAlgebra will have evolved from an outline of a program into one instantiation of a broader paradigm: **AI-native mathematics**, where definitions, lemmas, proofs and computational experiments live in a shared formal ecosystem accessible to both humans and machines.

The open question is not whether AI will change mathematics, but how mathematicians will choose to use—and be changed by—these new partners in proof.

References

[1] P. Chojecki. DeepAlgebra: an outline of a program. *arXiv:1610.01044*, 2016.

[2] The mathlib Community. The Lean mathematical library. *arXiv:1910.09336*, 2019.

[3] T. Hubert et al. Olympiad-level formal mathematical reasoning with reinforcement learning. *Nature*, 2025.

[4] Google DeepMind. AI solves IMO problems at silver-medal level. DeepMind Blog, July 2024.

[5] Google DeepMind. Accelerating discovery with the AI for Math Initiative. DeepMind Blog, October 2025.

[6] H. Xin et al. DeepSeek-Prover: Advancing theorem proving in LLMs through large-scale synthetic data. *arXiv:2405.14333*, 2024.

[7] Z. Shao et al. DeepSeekMath-V2: Towards self-verifiable mathematical reasoning. *arXiv:2511.22570*, 2025.

[8] D. Yin and J. Gao. Generating millions of Lean theorems with proofs by exploring state transition graphs. *arXiv:2503.04772*, 2025.

[9] M. Del Tredici et al. Ax-Prover: A deep reasoning agentic framework for theorem proving in mathematics and quantum physics. *arXiv:2510.12787*, 2025.

[10] Axiomatic AI. Ax-Prover benchmark results on AbstractAlgebra and QuantumTheorems. Technical report and benchmark page, 2025.

[11] X. Hu et al. StepProof: Step-by-step verification of natural language mathematical proofs. *arXiv:2506.10558*, 2025.

[12] DARPA. Exponentiating Mathematics (expMath). Program description, 2025.

[13] MIT News. MIT affiliates win AI for Math grants to accelerate mathematical discovery. 22 September 2025.

[14] Lean Forward Project. Formalizing algebra and linking LMFDB to Lean. Project description, 2024–2025.

[15] Y. Yang et al. LeanDojo: A benchmark suite for AI-assisted theorem proving. Preprint, 2023.

[16] 36Kr Europe. AI solved a 30-year math problem in just 6 hours. News article on Harmonic's Aristotle, November 2025.

[17] Mindplex Magazine. Harmonic's AI Aristotle claims solution to historic math puzzle. December 2025.

[18] Reuters. Robinhood CEO's math-focused AI startup Harmonic valued at $1.45 billion in latest fundraising. 25 November 2025.

[19] Hacker News. AI just proved Erdős Problem #124. Discussion thread, November 2025.

[20] B. Naskrecki and K. Ono. Mathematical discovery in the age of artificial intelligence. *Nature Physics*, 21, 1504–1506, 2025.

[21] A. Fawzi et al. Discovering faster matrix multiplication algorithms with reinforcement learning. *Nature*, 610, 47–53, 2022.

[22] B. Romera-Paredes et al. Mathematical discoveries from program search with large language models. *Nature*, 624, 70–76, 2023.

[23] T. H. Trinh et al. Solving Olympiad geometry without human demonstrations. *Nature*, 2024.

[24] Z. Azerbayev, B. Piotrowski, H. Schoelkopf, E. W. Ayers, D. Radev and J. Avigad. ProofNet: Autoformalizing and formally proving undergraduate-level mathematics. *arXiv:2302.12433*, 2023.