2k summary of "The Shape of Math To Come"

In an insightful and forward-looking essay titled *The Shape of Math To Come* (arXiv:2510.15924v1), mathematician Alex Kontorovich (Rutgers) presents a comprehensive analysis of how artificial intelligence and formal verification systems are poised to transform mathematical research. Writing from the perspective of September 2025, Kontorovich navigates the fundamental tension between the probabilistic nature of modern AI and the deterministic requirements of mathematical proof, offering a roadmap—quasi-autoformalization—for a future where mathematics is both accelerated by automation and grounded in unassailable rigor.

The paper opens by identifying a critical vulnerability in research areas like analytic number theory, which rely on long, technical arguments spanning scores of pages. In these intricate proofs, a single minor error can invalidate the entire result. While computer algebra systems (like Mathematica or Sage) can verify complex algebraic manipulations, Kontorovich sought analogous software to track the logical dependencies and constraints across an entire paper.

This search leads to Interactive Theorem Provers (ITPs), such as Lean. ITPs, in principle, allow mathematicians to "formalize" their work, entering proofs line-by-line until the computer certifies the argument's correctness down to foundational axioms. The reality, however, is sobering. Kontorovich notes that the existing formal libraries (like Mathlib, the primary library for Lean) are vastly underdeveloped. Currently, he cannot even formally *state* most theorems of interest, much less prove them.

This gap leads to what Kontorovich identifies as the fundamental conundrum, expressed in the inequality $\delta > \varepsilon$. New mathematics (δ) appears to grow exponentially, and formal libraries (ε) also grow fast, but currently, the creation of new math significantly outpaces the ability to formalize it. If this inequality persists, formalization will never catch up, and the dream of software as useful to reasoning as computer algebra systems are to calculation will remain a distant mirage. The only solution, Kontorovich argues, is to supercharge formalization with automation—requiring AI to do sufficient formalization to ultimately do research math.

The stochastic vs. deterministic divide

Kontorovich dedicates significant attention to the nature of Large Language Models (LLMs) and why they are fundamentally challenging tools for mathematics, despite their recent successes. He clarifies that an LLM is a function whose output is not the next word, but rather a *probability distribution* over all possible next tokens. The chatbots built atop these models make stochastic choices at every single step.

This stochasticity is the system's strength for versatile tasks but a profound weakness for mathematics, which requires deterministic correctness. A 99% correctly proved theorem is meaningless. Kontorovich illustrates this quantitatively: if an LLM achieves 99% accuracy at each

individual step, the probability of success for an argument requiring 1000 sequential steps is 0.99^{1000}, approximately 0.004%. A process that seems highly reliable at each step becomes almost certainly wrong when many steps compound. This relates to Daniel Kahneman's concepts of System I (fast, automatic) and System II (slow, deliberate) thinking; mathematical reasoning involves chaining many System I steps, where *every* link must be correct.

He acknowledges the rapid advancement of AI capabilities, citing the International Mathematical Olympiad (IMO). In 2023, no AI system could earn a single point; by 2025, multiple systems claimed gold medal scores. This suggests that perhaps scaling compute and data will eventually allow LLMs to sustain longer, reliable chains of reasoning.

However, Kontorovich argues that even perfect scaling is insufficient. He imagines a future AI system that generates 100 research papers per day, with 99 being completely correct and only one containing a subtle error. Would a mathematician rely on such a system? Kontorovich asserts he would not. To use the tool effectively, he would need to verify all 100 papers to find the flawed one. Since verifying cutting-edge research is often as difficult as producing it, the automation provides little practical benefit. The theorems themselves could not be treated as established facts.

This is not a problem scaling can solve. Even at 99.999% accuracy, a non-zero error rate creates an unusable tool if the correct results cannot be distinguished from the flawed ones without manual verification. He contrasts this with a hypothetical AI that produces only 10 papers per year, but each comes with a *formally verified* proof. This is a tool he would use enthusiastically. The difference is deterministic certification. Quality and certainty matter far more than quantity when building a cumulative body of mathematical knowledge.

A path forward: quasi-autoformalization

The bottleneck remains the gap between natural language mathematics and formal libraries ($\delta > \varepsilon$). AI assistance is essential, not primarily to prove new theorems, but for *autoformalization*, that is, automatically converting established mathematics into formal proofs to accelerate library growth.

Kontorovich proposes a framework called *quasi-autoformalization*, a system of cooperating AI agents orchestrated to translate natural language mathematics into formal proofs. The key word is "quasi": he does not require full autonomy but rather a collaborative and accelerated workflow where human intervention is welcome to guide the process.

He outlines an architecture consisting of four agents:

The Decomposer: Natural language proofs are written at the level of "Ideas," hiding steps obvious to humans but necessary for formal completeness. The Decomposer breaks down these ideas into sufficiently small, explicit steps suitable for formalization.

The Translator: This agent converts the natural language steps into formal Lean code. It produces a "scaffold" of the theorem, using intermediate claims (have statements) that are initially *sorry'd out* (meaning their proofs are postponed). n Lean, `sorry` is a standard placeholder tactic. It tells the theorem prover to accept the current statement as true for the moment, allowing the user to focus on the high-level proof structure before filling in the details.

The Solver (or "Closer"): This agent attempts to prove each have statement, replacing sorrys with complete formal proofs using automated theorem-proving techniques and searching through Mathlib.

The Conductor: This agent orchestrates the process. If a formalization fails, the Conductor diagnoses why: Is the goal too complex (requiring further decomposition)? Was there a mistranslation? Does the result already exist in Mathlib (using Retrieval Augmented Generation - RAG)? Recall that RAG is a technique that enhances the language model by granting it access to a specific, external knowledge base—in this case, the Mathlib repository. Instead of relying solely on training data (which may be outdated or prone to hallucinating non-existent lemmas), the system actively searches the current Mathlib codebase. This allows the Conductor to retrieve precise, compilable definitions and existing theorems to augment its next generation step, ensuring the formalization is grounded in the actual library.

Kontorovich recounts optimistic experiments with state-of-the-art systems like DeepMind's AlphaProof. He describes how AlphaProof brilliantly closed a seemingly trivial but formally complex goal regarding the Riemann zeta function, despite having only been trained on high school IMO problems. These successes highlight the potential of the approach, but significant challenges remain.

Difficulties and robustness

The process is rarely straightforward. One key challenge is the subtlety of translation. Getting formal statements to say exactly what is intended is sometimes as hard as proving them.

Kontorovich illustrates this with the concept of "total functions" in Lean. In most ITPs, every function must be defined for all inputs. For example, the expression $1/0$ must have a "junk value"; in Mathlib, it is assigned the value of 0. This simplifies many proofs but requires careful handling. He notes that if the Riemann zeta function's "junk value" at its pole were zero, the standard statement of the Riemann Hypothesis would be trivially false. This illustrates just how treacherous translation can be.

When Kontorovich discussed these difficulties with Christian Szegedy (a leader in autoformalization), Szegedy offered a profound response: "It doesn't matter if the formalized statements and definitions are wrong," because mathematics, in practice, is *robust* and self-correcting. Definitions and theorems emerge through refinement. Wrong statements that are useful will be discovered and corrected through use.

Kontorovich, however, remains cautious, feeling mathematicians must hold themselves personally responsible for formal statements. He cites a troubling example where an AI model, asked to compute a result using an external tool (Sage), *pretended* to execute the code and reported its best guess when tool access was accidentally disabled. This illustrates why, even when an AI claims to use formal tools, blind trust is impossible.

The formalization factor and adoption

If this vision pans out and Mathlib grows sufficiently large (and robust), when will the broader mathematical community adopt these tools? Kontorovich draws a parallel to the adoption of LaTeX. In the past, self-typesetting was cumbersome. As LaTeX became efficient, it also served as an organizational tool for the research process itself. He defines the *Knuth factor* as the ratio of time required to develop and document a result using LaTeX versus by hand. When this factor dropped below 1 (around 1990), nearly everyone switched voluntarily because it sped up their workflow.

He expects the same for Lean. He critiques the traditional de *Bruijn factor* (ratio of lines of formal code to lines of natural language proof), arguing that lines of code are no longer a meaningful proxy for human effort due to AI assistance.

Kontorovich proposes a new metric: the *Formalization Factor*. This is the ratio of time required to discover and formalize a result (from idea to verified proof) to the time required to discover and write up the same result in LaTeX. Currently, this factor is well above 1 (perhaps 10 or 100) because the foundational libraries are incomplete. But once it drops below 1, mathematicians will switch voluntarily to speed up their workflow and increase confidence in their results. For this to happen, several conditions must be met: Mathlib must become comprehensive (requiring AI assistance so that $\varepsilon > \delta$); the tools must be accessible (like a browser-based "Overleaf for Lean"); and AI must handle the tedious parts, allowing mathematicians to focus on the creative work. He predicts this could happen within 5-10 years for many core areas, provided sustained progress in AI-assisted formalization.

Transforming teaching and understanding

The implications extend beyond research efficiency; teaching represents a parallel and potentially accelerating force. Kontorovich uses a powerful analogy: imagine teaching chess purely by describing moves in algebraic notation (1. Nf3 Nf6...). Chess aficionados can follow this, but for beginners, it is vastly easier to just *show the board*. Yet, this is precisely how mathematics is currently taught. At every step of a natural language proof, the "game board"—the current objects, assumptions, and goal—is changing. Professionals track this effortlessly, but beginners struggle. When teaching with Lean, the game board is always there, automatically generated and continuously updated. Beginners don't have to reconstruct it from terse prose; they can simply look.

Finally, formalization may fundamentally transform mathematical understanding and communication. Kontorovich envisions a shift from the current linear and static presentation of

proofs. Formal systems could enable "proof at multiple resolutions"—interactive arguments that adapt to the reader's expertise, allowing them to drill down from high-level intuition to granular formal steps as needed. This could also enable a profound shift in *mathematical responsibility*. Currently, accountability is intensely "vertical"; mathematicians feel responsible for understanding the proof of every theorem they cite. Formalization could enable genuine "modular mathematics." If formal statements specify precisely what they assume and guarantee, mathematicians could confidently use off-the-shelf components, similar to engineers using standardized parts. This transition from vertical responsibility to horizontal collaboration could dramatically accelerate mathematical progress.

Conclusion

The vision outlined—of quasi-autoformalization accelerating library growth, formalization factors dropping below unity, and mathematical practice migrating toward formal systems—represents both tremendous opportunity and considerable uncertainty. The transformation depends on resolving the fundamental tension between the stochastic nature of current AI and the deterministic requirements of mathematical proof. Kontorovich offers an optimistic perspective on a potential Golden Age of human-computer collaboration. He suggests AI assistance may help discover hidden low-hanging fruit—problems that seem intractable but are actually straightforward when approached with the right combination of techniques, explored via AI and guaranteed by formal verification.

The shape of mathematics to come, he concludes, depends on the choices made today. If we prioritize pure automation over human insight, we risk creating systems that solve problems without advancing understanding. But if we succeed in building AI that amplifies mathematical intuition—handling the mechanical aspects while preserving human creativity—we may witness not the end of pure mathematics, but its transformation into something more powerful and beautiful. The mathematical community stands at a remarkable inflection point, living through a pivotal moment in the history of mathematical reasoning.