Mathematical Reasoning Agents

From Rethlas to Danus

Shurui Liu
Stanford University

Cursor · July 2026

Why AI for research mathematics

AI already performs strongly on mathematical competitions, including the IMO and Putnam. Why care about AI for mathematical *research*?

- **Personal purpose.** Mathematicians care about research problems.
- **Open-ended problems, unlimited/unknown scope.** Competition problems have solutions within a fixed syllabus and often share techniques — like hard LeetCode problems (standard libraries, known algorithm templates, guaranteed to be doable). Real research — like real industry problems — is rarely like that.
- **Long horizon.** A much larger "search space": experiments take hours, days, or even weeks; papers run 5, 20, 100 pages or more.
- **Other real issues.** No textbooks for the most recent frontier; internet resources (arXiv, MathOverflow, even refereed papers) may contain errors and misguide agents.

Everything that makes math hard for agents is a concentrated version of what makes long-horizon software tasks hard.

Plan for today

Two math agents:

1. **Rethlas** — a linear generate–verify loop
2. **Danus** — orchestration of multiple agents: an organization

Focus:

1. the intuition and considerations behind the system designs
2. transferable principles and ideas for building agentic frameworks

Mimic the human workflow

Skills, tools, roles, memory

What does a mathematician do with a hard problem? Never "just prove it":

- construct **toy examples**
- hunt for **counterexamples**
- **search** the literature
- propose **decomposition plans**
- **direct proving** (screen each plan)
- **recursive proving** (fan out)
- **identify key failures**

- Each habit becomes an explicit, selectable **skill**; the generation agent loops *assess state* $\rightarrow$ *choose a skill* $\rightarrow$ *act*.
- Competing strategies are juggled the way a person juggles them: **one subagent per decomposition plan**, in parallel, each seeded with the *stuck points of the other plans*; if all fail, the common obstructions seed the next round of plans.

- Mathematicians need literature search; the agent gets **Matlas** (`matlas.ai`) — a semantic theorem search engine: **8.51 million statements** from 435,000 published papers across 180 journals and 1,900+ textbooks.
- Retrieved statements come with extracted dependencies and hierarchical unfolding — self-contained results, not pages of prose.
- A detail motivated by experiments and by human workflow — **think, then look it up**: odd iterations of the loop *block* search entirely (reason from what you have); even iterations re-enable it.

Skills mimic habits; tools mimic resources.

CONTEXT ENGINEERING

Memory management

Think from agentic viewpoint: context

The loop runs for hours — it needs a mathematician's notebook:

- **Scratch paper:** the raw run logs — cheap, voluminous, disposable.
- **Structured notes:** append-only JSONL channels per problem — `toy_examples`, `counterexamples`, `subgoals`, `failed_paths`, `big_decisions`, `verification_reports`, `...` — retrieved by BM25.
- **Failed paths are mandatory entries.** Every dead end is written down and stays queryable — no repeating failures.

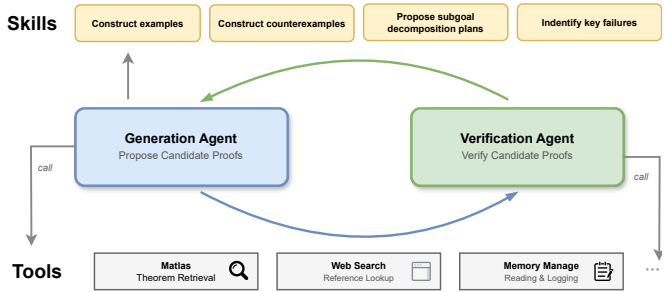
Verification

- We need an assessment of correctness — but the author tends to be optimistic when grading its own homework.
- Humans fail similarly: authors believe their own proofs, and subtle gaps survive proofreading by the person who wrote them.
- **Human viewpoint:** the mathematical community's answer is centuries old — **the referee**, someone who did not write the proof and does not need it to be true.
- **Context viewpoint:** the verifier should be a fresh agent, carrying no memory or context from the generation process beyond the task itself.

CONTEXT ENGINEERING

Separation of roles

Lever two: what each role is exposed to — no agent both creates a proof and certifies it.



- **The generation agent keeps continuity**: one conversation, resumed across iterations — intuition, partial progress, and known dead ends persist.
- **The verification agent gets fresh eyes, by construction**: a stateless HTTP microservice spawning a *fresh process per submission* — zero shared context with the prover. It cannot inherit the author's optimism, because it cannot see it.
- Blueprint in $\rightarrow$ strict JSON verdict + repair hints out $\rightarrow$ revise; accept only at **zero critical errors and zero gaps**.

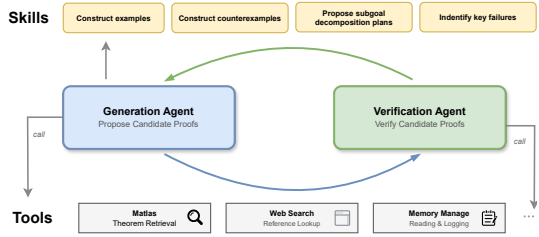
Role separation separates not only textual context but also tools and skills — it **decomposes the job**. The verifier gets its own craft:

- Split the proof into the **smallest deduction steps**; check each for logical validity, theorem applicability, missing or redundant hypotheses.
- **Reference correctness**: does the cited theorem exist — and does its *statement* say what this proof needs, in this context, with these hypotheses?
- **Synthesize a strict report**: aggregate every error and gap, apply the accept/reject rule, and produce repair hints when rejecting.

Two roles, two skill sets, two context policies

What we built: Rethlas in one slide

- Two agents and a loop: a **generation agent** (skills = research primitives, tool = Matlas, memory = scratch paper + structured notes) writes an informal proof *blueprint.md*; a **verification agent** (skills = referee primitives, tool = Matlas, memory = none) referees it; verdict and repair hints flow back until acceptance.
- Built as coding-agent CLIs (Codex / Claude Code) under a supervisor script; the paper's results ran on GPT-5.4 / GPT-5.5.
- A diligent mathematician is expected to find a solution whenever one exists in the search space, by trying persistently and rejecting false paths. The generate–verify loop spends compute to chase the same dream.



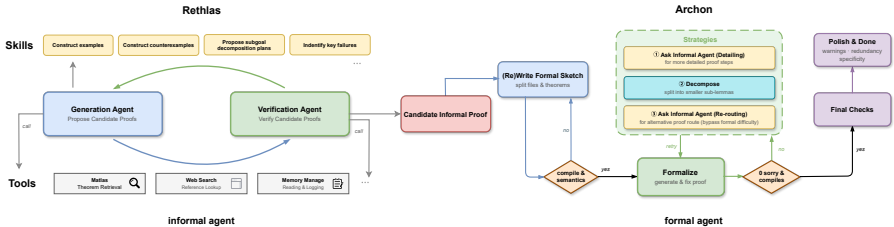
Case study 1: an open problem of D. D. Anderson (2014)

Problem (Anderson, 2014)

Does weak quasi-completeness imply quasi-completeness for Noetherian local rings?

- Open for a decade; Rethlas, tasked with *prove or refute*, found a **counterexample after ∼45 minutes** of autonomous reasoning.
- The key ingredient — a 2006 result of Jensen from the distant generic-formal-fiber literature — was **surfaced by Matlas**: the kind of cross-field connection that ordinarily requires experts from two fields meeting.
- The result was then formalized in **Lean 4** by Archon, our companion formalization agent — 19,448 lines across 42 files, built autonomously in ∼80 hours. (Formal methods provide a strictly correct verifier; this talk focuses on the informal reasoning agent.)

From informal to formal: the Rethlas–Archon pipeline



Rethlas's generate–verify loop (left) produces the informal proof; Archon (right) compiles it into a kernel-checked Lean 4 project — informal exploration and formal certainty, integrated end to end.

Case study 2: an open research problem in algebraic geometry

Rational conjugacy classes of 1-parameter subgroups

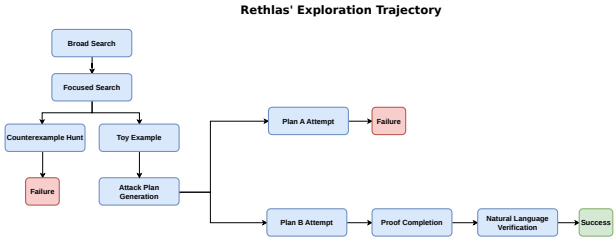
Let G be a connected smooth affine group scheme over a field k, and let $\mathbb{X}_*(G) = \mathrm{Hom}_{k\text{-group}}(\mathrm{GL}_1, G)$ with $G(k)$ acting by conjugation. For a field extension K/k, is

$$\mathbb{X}_*(G)/G(k) \longrightarrow \mathbb{X}_*(G_K)/G(K)$$

injective?

- Research-level algebraic geometry (a lemma/proposition), recorded nowhere in the literature or on the internet.
- Three easier cases (split reductive, reductive, perfect field) are known to experts; both Rethlas and GPT-5.5 Pro can do those.
- The general case may not be obvious to experts. GPT-5.5 Pro: a completely wrong proof, with almost no mathematical progress. Rethlas: a correct proof in 44 minutes.

What went right, and the one thing that went wrong



Only one tiny mistake — a **citation-substitution error**: lacking the paywalled book, it cited a survey theorem A whose *proof* points back to the needed result B, but whose *statement* does not literally contain it.

The agent is not fully to blame:

- **Copyright:** the needed result lives in a copyrighted book; the logs show the agent knew the book and the theorem number, but could not retrieve the statement.
- **Skill design:** we require the agent to read proofs of a statement and extract insights and techniques — and here the proof of theorem A does point to result B.

More results, across fields

Area	Results
Algebraic geometry	1-parameter subgroups; p-adic Simpson lift-independence; foliations (bend-and-break, Shokurov index); klt-type questions
Commutative algebra	Anderson 8a and five more problems from the CFFG collection; Boij–Söderberg Questions 6.1/6.2 (Erman–Sam)
Analysis	Brezis Problems 5.1 and 5.3 (Sobolev circle maps)
Probability	Vinzant's refined injectivity conjecture (a)

`frenzymath/Rethlas_results`

A partial, growing list — each result checked by human experts before release.

Limitations and scaling

The linear loop oversimplifies the workflow of mathematicians:

1. a mathematician accumulates *many small facts* and constantly adjusts strategy, guided by intuition about the facts obtained so far;
2. real projects often need *collaborators* with different expertise, trying different approaches at the same time and communicating.

The CS viewpoint:

1. **Context:** the solution is bound to the generation agent's context window; verified partial results *and their proofs* are reloaded every iteration, even when only the statements are needed.
2. **Search trajectory:** purely DFS; it should mix BFS and DFS, dynamically choosing focus and strategy — especially when the dependency structure is complicated (not linear).
3. **No parallelism.**

Limitations and scaling

Rethlas cannot be naively scaled or parallelized — structurally:

- **Input: one hypothesis at a time.** The loop commits to a single direction; breadth exists only inside one agent's context.
- **Context: every agent carries everything.** Compaction helps, but at minimum the deliverable *blueprint.md* and the structured notes are loaded every iteration.
- **Output: concurrent edits collide on the shared whiteboard.** Two agents cannot safely improve the same Markdown document.

Multiple agents are needed, and the deliverable must be "modular":
shared truth, division of labor, and rules about who may do what.

Mimic the human workflow

This time the unit is not a mathematician — it is a research group.

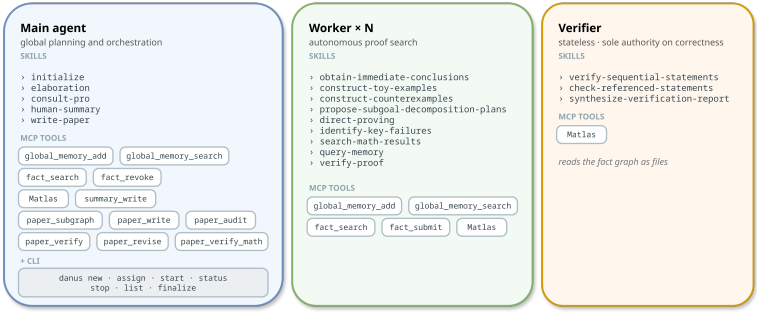
Who works on what? The same way a group decides:

- **Group leader** (main agent): plans, assigns, monitors, writes reports — and *does no math*.
- **Workers** (Rethlas generation agent): each owns one claim at a time; run at two effort tiers *on purpose* — lower effort buys breadth and diversity, higher effort buys depth.
- **Senior expert** (GPT-5.5-pro): consulted at most about once an hour, on distilled state — strategy, not labor. *Spend your most expensive intelligence at low frequency.*
- **The group meeting** (strategy loop): elaborate an honest synthesis → consult → record `master_guidance` → per-worker `TASK.md`. Workers never talk to each other — everything flows through the shared stores.

CONTEXT ENGINEERING

Separation of roles

At scale, separation of roles becomes separation of powers.



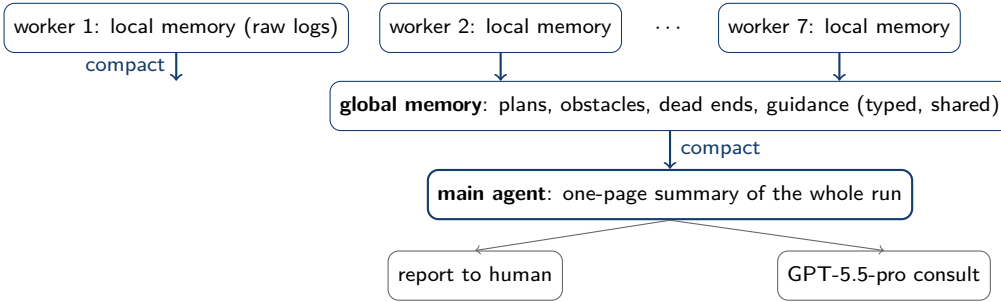
- **Role gating**: the MCP server exposes a *different tool subset per role*. Workers have `fact_submit`; the main agent *has no submission tool at all*; the verifier only reads.

CONTEXT ENGINEERING

Memory management

Three-tier memory: local, global, and the fact-graph DAG.

- Real research is the **accumulation of many small facts** — so a worker submits **one claim at a time** (a lemma, a counterexample, a toy computation), judged by a *fresh verifier instance*.
- **Fact graph**: accepted facts enter the fact graph (**DAG**: node = fact ID + statement + checked proof; edge = logical dependency) — the system's *unique source of truth*.
- **Global memory**: everything else — plans, dead ends, master guidance — lives in global memory, which is **explicitly not truth**.
- **Local memory**: each worker keeps its own scratch paper, as in Rethlas, and receives the verifier's feedback.



- **Width:** the proof decomposes into units individual workers can *own* — parallel contributions accumulate instead of colliding, and each worker loads *only the facts it needs*.
- **Depth: compaction at every boundary** — local logs are compacted into global memory, global memory into the main agent's summaries; the main agent reads compacted state, never raw logs — chains of reasoning far beyond one context window.
- **Non-linear dependency structure:** a worker can search and load verified results from other directions, and hence may integrate different routes.

A new failure mode: the organization deceives itself

Individual overconfidence, we solved with a referee. Organizations fail in new ways — above all through the main agent's over-optimism:

- **Progress reports drift optimistic** — and each strategic cycle summarizes the last, so small optimism *compounds*.
- **The writer invents at the seams.** When the main agent stitches many verified facts into one linear paper, the connecting prose *between* facts can introduce new, unverified claims — and a citation written from memory instead of from a record can simply be made up.
- **Effort feels like progress.** With seven agents always busy, when is the work actually *done*?
- **One bad fact poisons everything downstream** — in a deep dependency chain, a single false acceptance is not a local error.

Honesty cannot be requested. It has to be designed in — at every level.

Design against self-deception

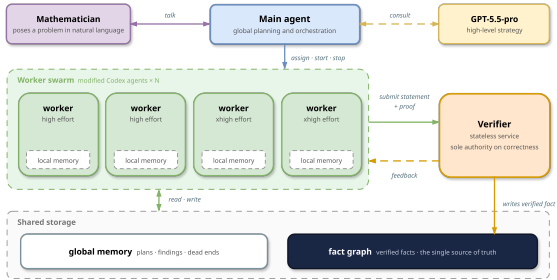
An organization can lie to itself in ways an individual cannot.

- **Verifier reliability is load-bearing.** With ∼500 facts chained into one theorem, even a 1% per-fact error rate leaves the whole chain sound with probability below 1% (0.99^{499}). The verifier is engineered — and measured — to produce *essentially no false positives*.
- **Revocation cascades**: when a wrong fact is nevertheless found, it is removed together with every transitive dependent — needed twice across all experiments.
- **Optimism is banned structurally**: the summarization skill is written as *prohibitions* — no numeric progress estimates, no "almost done."
- **Citations are data**: captured structurally at proof time; the bibliography is assembled from those records — hallucinated references are prevented *by construction*.
- **Termination is result-gated**: the run stops when the target theorem exists as a verified fact — not when effort feels sufficient.

Fully faithful vs. rigorous vs. human-readable:

- An honest fact-by-fact translation of the fact graph is *not* human-readable: a paper needs motivation, omits proofs of routine results, and is organized section by section.
- Left alone, a writing agent may omit too many details, or coin unprofessional terms ("firewalls," "inductive engine," . . .).
- **Our solution: the writer must face the verifier.** The main agent writes the paper — doing no math, only retrieving facts from the fact graph — and the draft must *pass the verifier*: a generate–verify loop for writing.
- Remaining issues — unconventional notation, unconventional citation formats, over-omitted details — but these are much easier for human experts to spot and fix.

What we built: Danus in one slide



- A **main agent** (Claude Opus 4.8 on Claude Code) plans, coordinates, and reports — no math, no submission tool. **3–9 workers** (Codex / GPT-5.5) prove one claim at a time; a **stateless verifier** is the sole gate into the **fact graph**.
- Rethlas survives inside as a single worker–verifier loop; the worker and verifier skills are inherited essentially unchanged. *What changed is the orchestration.*

Orchestration is the variable

Same models, same prompt, same verifier — a controlled experiment.

Case study: tangent classes of matroid building sets

A research problem posed by Ronnie Cheng (Stanford math PhD student), with his own preprint *deliberately withheld* from all systems — the problem they faced was open. The raw prompt, verbatim:

```
Let $M$ be a loopless matroid (not necessarily realizable) of rank $d + 1$
on the ground set $E$, and let $G$ be a building set containing the top
flat $E$. The task is to construct a tangent class $T_{M,G} \in K(M,G)$
such that
(i) If $M$ is realizable by $L$, $T_{M,G}$ is the tangent class for the
wonderful compactification $W_{L, G}$, which is a projective variety of
dimension $d$.
(ii) For integer $i$,
$\dim_{\mathbb{Q}} A^i(M, G)=(-1)^i\chi_{M, G} (\wedge^i T_{M, G}^\vee)
= (-1)^i \deg_{M, G}(ch(\wedge^i T_{M, G}^\vee) \cdot td(T_{M, G}))$
(iii) For $k \leq d$, we have
$\deg_{M,\G} (c_k(T_{M,\G}) \alpha^{d-k}) \geq \binom{d+1}{k}$
and prove rigorously that the constructed class satisfies the above
properties.
```

Danus re-produced the **main results** of Ronnie's paper (arXiv:2606.22650); the experiment paper (Cheng–Liu, arXiv:2607.05835) is on arXiv.

The run, in numbers

- **7 workers** (3 xhigh, 4 high), ∼5 days wall-clock, ∼50 h active runtime
- **13** consultations of GPT-5.5-pro
- **3,157 verified facts**, 8,616 dependency edges
- dependency chains up to **54 deep**
- **499** facts form the theorem's supporting closure
- global memory: 636 proof attempts, 151 counterexamples, 25 dead ends

Clusters = separate lines of attack (abandoned scaffolding, an independent Chern-bound re-derivation, the integral extension). **The theory built is** 6× **larger than the surviving proof** — what exploration costs, and buys.

The controlled comparison — read it as an ablation

Same verbatim prompt, no suggested route, preprint withheld from all three:

System	What it has	Outcome
GPT-5.5-pro (web agent)	a frontier model	no solution — outline already wrong
Rethlas, three runs	the loop	0 / 3 passed the verifier
Danus	the organization	verified solution

- Rethlas got stuck on subtleties in algebraic geometry (realizable matroid vs. wonderful model vs. toric variety) — exactly what parallel exploration plus fact-level verification untangles.
- Honesty: one gap survived to human review (a lemma treated as proved without a valid argument — true, with a short proof).

Human–AI collaboration

Humans supply direction; the machine supplies verified execution — and its own repairs.

- **Shokurov index conjecture for foliations (dim. 3):** Danus split the problem into five classes on its own and solved three via a Lie-theoretic route the project's experts had not anticipated — the three *hard* ones, in those experts' judgment. It stalled on the two cases humans find easy, until a single hint — "foliated minimal model program" — unblocked them.
- ***p*-adic Hodge theory (Rethlas):** proved the proposed conjecture, exposed a redundant hypothesis, sharpened "genus large enough" into an explicit bound — and, asked whether an assumption could be dropped, built a counterexample from a field so distant the authors said they might never have found it.
- When an error was found, the repair was always carried out by the system, never the human. "Like explaining a proof to a graduate student: indicate difficulties, point out errors, supply references — without spelling out the details."

A rough math $\leftrightarrow$ CS dictionary

Writing a research paper is not the end of this journey. A way to size a mathematical task:

Math	CS
math olympiad	LeetCode
lemma / proposition / short paper	a function / a short API
main theorem of a paper	an API
novel method	a new algorithm
theory building	a new language (C++ $\rightarrow$ Rust)

Today's systems reach roughly the upper rows; theory building — the bottom row — is the open frontier.

Work in progress

There are several ongoing math collabrations originating from interaction with Rethlas-Danus. But the system is far away from being prefect:

- **Simplifying the harness for the next model generation** (GPT-5.6Sol, Claude Fable 5).
- **Creativity**: still no theory building; still within the convext closure of human knowledge. The verifier is strict, so workers take tiny steps; non-rigorous exploration never happens, which may punish novel ideas.

What may transfer — the principles

- **Mimic the human workflow.** Distill the practitioner's habits into skills; at scale, mimic the *organization*, not the individual.
- **Context engineering.** Control what each agent sees, with two levers:
  - **memory management** — scratch paper → structured notes → modular/nonlinear deliverable (DAG); compact at every boundary; keep the dead ends; reset context if applicable
  - **separation of roles** — each role its own context and tools: judges get fresh eyes, workers get bounded reads — enforced with capabilities or deterministic code
- **Slogan:** a well-organized loop with a solid verifier, clean context, and layered memory meaningfully buys compute and searches the solution space effectively — at the least, solving problems in the *convex closure of human knowledge*.

Challenges in general:

1. **Debugging agentic systems**: 30% → 80% capability is easy; 80% → 90% is hard; 90% → 99% is extremely hard. Benchmarking/evaluating is very emperical.
2. **Effort engineering?** The boundary of this power?

References

Systems

- H. Ju, G. Gao, J. Jiang, B. Wu, Z. Sun, S. Liu, et al. *Automated Conjecture Resolution with Formal Verification* (Rethlas + Archon). arXiv:2604.03789. Code: `github.com/frenzymath/{Rethlas, Archon}`; results: `frenzymath/Rethlas_results`.
- J. Liu, G. Gao, Z. Sun, B. Wu, S. Liu, J. Jiang, H. Ju, S. Qin, R. Hu, L. Chen, R. Cheng, B. Dong. *Danus: Orchestrating Mathematical Reasoning Agents with Fact-Graph Memory*. arXiv:2607.06447. Code: `github.com/frenzymath/Danus`.
- Matlas, a semantic search engine for mathematics: `matlas.ai`.

Math Results mentioned today

- R. Cheng, S. Liu. *Tangent classes of matroids and wonderful compactifications* (appendix by G. Gao and S. Liu). arXiv:2607.05835.
- R. Cheng. *Tangent classes for matroid building sets*. arXiv:2606.22650.
- X. Pan, J. Yu (p-adic Simpson, arXiv:2605.29947); J. Jiang et al. (commutative algebra, arXiv:2605.25259); X. Dou, Z. Jin (Brezis problems, arXiv:2605.24626); Z. Li (phase retrieval, arXiv:2606.17922).

Thank you!