Recurrence: An Evolving Research-State Copilot for Mathematics

Jun Li [1]

Abstract

Much of recent AI-for-math frames theorem proving as a single proof obligation: given a statement, generate, check, and revise a proof. In mathematical research, however, progress often depends on the surrounding state that comes before and between proof attempts: fixing the right statement, testing examples and counterexamples, remembering failed routes, extracting reusable lemmas, and deciding what should be attacked next. We present Recurrence, a self-evolving copilot for this layer of mathematical work. Recurrence adds a pre-proof layer to a proof-and-verification pipeline: it locks the target statement, separates exploratory mapping from direct attack, keeps proof-side and disproof-side routes visible, labels claims through adversarial verification, and stores memory items that guide later runs. Its main output is not only a proof attempt, but a reusable record of the proof attempt, counterexample search, verifier objections, failed routes, remaining bottlenecks, and the recommended next target. By "self-evolving" we mean non-parametric, process-level evolution: the research state, prompts, and memory evolve across runs; no model weights are updated. We report pilot evidence from topology campaigns and a cluster-benchmark protocol instantiated on a 22-target Gompf problem list, showing how such artifacts preserve obstructions, avoid false solution claims, and produce human-checkable next steps. Code: https://github.com/Recurrence-Poincare/Recurrence-Agent.

[1]California State University, Northridge, CA, USA. Correspondence to: Jun Li <jun.li@csun.edu>.

The 3rd AI for Math Workshop at the 43rd International Conference on Machine Learning (ICML), Seoul, South Korea, 2026. Copyright 2026 by the author(s).

1. Introduction

AI-for-math systems increasingly treat theorem proving as an agentic proof-generation problem: given a statement, generate a proof, verify or critique it, and revise. Recent systems and benchmarks, including Aletheia, QED, Rethlas/Archon, Aristotle, AlphaGeometry, and FormalProofBench, show rapid progress on this prover and formalizer layer (Feng et al., 2026b; An et al., 2026; Ju et al., 2026; Harmonic, 2025; Trinh et al., 2024; Ravi et al., 2026). Recurrence targets a complementary layer of mathematical work. In research practice, the hard part is often not only producing the final proof text, but maintaining a stable state across related problems: exact statements, examples and counterexamples, failed routes, reusable reductions, verifier objections, and judgments about what to try next.

We use research state to mean the durable, inspectable record of where a mathematical investigation currently stands: the locked target statements, active proof and disproof routes, examples, obstructions, failed attempts, verifier judgments, reusable lemmas, and prioritized next steps. The missing object is not another generated proof; it is a durable research-state object. Recurrence is built around this premise. It is a mathematician-in-the-loop copilot whose primary output is a reusable record of the exact claim, the routes explored, the failures found, and the next problem to inspect.

This framing is aligned with the theme of self-evolving scientific agents, but the evolution is deliberately modest and inspectable. The system does not update model weights. Instead, it updates the mathematical state: stabilized targets, branch ledgers, failed-route warnings, verifier-labeled claim maps, and next problems. Recent surveys define self-evolving agents broadly as systems whose model, context, tools, or architecture change in response to trajectories or feedback, while noting that full autonomy remains an aspirational goal and that proto-evolution includes feedback-driven prompting (Gao et al., 2026). Under this terminology, Recurrence is a proto-evolutionary scientific agent: it performs experience-dependent non-

parametric updates to memory and prompts, uses these updates to alter later search behavior, and includes an explicit Explore mode for autonomous branch generation.

This view is motivated by recent accounts of human mathematics as compressed structure. Aksenov et al. (2026) argue that definitions, lemmas, and theorems act like reusable macros that make a vast formal deduction space navigable. At the same time, mathematical progress is not only compression into formal proof. Poincare emphasized that logic and intuition play complementary roles in discovery (Poincare, 1905), and Tao describes mature mathematical practice as combining rigorous formalism with intuition and the big picture (Tao, 2007). In recent discussions of AI for mathematics, Tao's notion of proof digestion similarly shifts attention from merely generating or verifying a proof toward making it explainable, reusable, and connected to future work (Stanford University, 2026; Ghosal, 2026). Recurrence targets this intermediate layer: before full formalization is available, it turns exploratory attempts into inspectable research artifacts.

Before proof generation, Recurrence rephrases the problem without changing its meaning, separates hypotheses from conclusions, records quantifiers and success criteria, and chooses between two modes. In explore mode, the agent maps nearby positive and negative conjectures before committing proof budget. In attack mode, it treats the input as a serious conjecture and simultaneously enumerates proof methods and disproof methods. Both modes must "recur" to the original target: every branch is classified as direct progress, reduction, obstruction, finite evidence, side result, or dead end. Code and representative run artifacts are supplied through the project repository or supplementary archive.

The main contribution is a locked-statement research harness for long-horizon AI-assisted mathematics:

- locked-statement discipline for fragile mathematical targets;
- explore and attack modes that keep proof-side and disproof-side pressure visible;
- statement-drift and whole-proof verification that distinguish artifact honesty from theorem truth;
- failure ledgers, no-output decisions, claim maps, and return maps that convert failed attempts into reusable guardrails; and
- campaign memory for non-parametric process-level self-evolution: later runs inherit frontier lemmas, dead ends, prompt lessons, and verifier-labeled claim maps.

Table 1. Recurrence artifact contract.

Artifact	Required content	Prevents
Target card	objects, hypotheses, quantifiers	nearby problem drift
Branch ledger	positive/negative routes, status	unstructured wandering
No-output note	fatal gap or missing theorem	hallucinated success
Drift check	target comparison, category audit	theorem replacement
Claim map	lemma graph with labels	opaque proof blobs
Memory item	reusable warning or next target	repeated dead ends
Return map	effect on original conjecture	loss of target

We evaluate this through topology-centered pilot runs and a cluster-benchmark protocol instantiated on the Gompf problem list. This benchmark asks a narrower but measurable question: can the agent read a fresh cluster of open problems, avoid false solution claims, preserve exact statements, identify reusable subproblems and warnings, and output artifacts that working mathematicians can rate for usefulness?

We do not claim that Recurrence replaces formal verification or expert review. The prover and formalizer layer is increasingly crowded and predominantly model-centered. Recurrence focuses on a complementary copilot layer: human–AI co-evolution of the research process itself, where mathematicians can distill field-specific judgment, taste, and failure experience into prompts, artifacts, and memory that later runs inherit.

2. Locked-Statement Contract

Recurrence is distinguished less by any single agent role than by the contract imposed on its artifacts. A useful run must compress a transcript into a research record that a later human or agent can inspect and reuse. In particular, every branch receives a return-to-target status; every failed route is converted into a warning, obstruction, or next problem; every claimed lemma receives a verifier label Yes, No, or Uncertain; and Uncertain is never promoted to a final mathematical result.

3. Related Work

Recent systems show rapid progress at the prover and formalizer layer. Aletheia is a Gemini Deep Think-powered research agent for natural-language mathematics, with reported autonomous performance on FirstProof (Feng et al., 2026b;a). Rethlas/Archon combines an informal research agent with Lean formalization and theorem retrieval (Ju et al., 2026); Aristotle integrates informal reasoning, lemma generation, Lean proof search, and geometry solving (Harmonic, 2025); AlphaGeometry is a landmark neural-symbolic geometry prover (Trinh et al., 2024); and FormalProofBench shows that graduate-level formally verified proof generation remains difficult even for frontier models (Ravi et al., 2026). At the frontier, OpenAI recently reported that a general-purpose reasoning model produced a counterexample to Erdős's planar unit-distance conjecture; the proof was checked by external mathematicians and accompanied by a human-verified expository note (OpenAI, 2026; Alon et al., 2026). Finally, QED provides a useful open-source multi-agent proof-system baseline for attacking open mathematical problems (An et al., 2026).

The open prover ecosystem is also advancing quickly. The DeepSeek-Prover line uses large-scale synthetic Lean data, proof-assistant feedback, Monte-Carlo tree search, and subgoal decomposition to improve formal theorem proving (Xin et al., 2024a;b; Ren et al., 2025); DeepSeekMath-V2 complements this line by emphasizing self-verifiable theorem proving (Shao et al., 2025). Kimina-Prover and Numina-Lean-Agent show the strength of open, agentic Lean workflows driven by reinforcement learning, tool use, retrieval, and general coding agents (Wang et al., 2025; Liu et al., 2026). Axiom Math's ProofOptimizer addresses a complementary bottleneck: even machine-checked proofs may be too long or opaque for human mathematical use, so proof simplification becomes a bridge from verification to understanding (Gu et al., 2026).

Workshop-aligned benchmarks broaden the evaluation target beyond isolated pass rates. The AI4Math 2026 workshop emphasizes verification and measurement, human-AI collaboration, scientific agents, multimodal reasoning, and scientific problem solving (AI for Math Workshop, 2026). Its challenge tracks include semantic alignment for autoformalization, theoretical computer science proving in Lean with CSLib, visual physics reasoning with SeePhys Pro, and end-to-end Lean autoformalization (Barrett et al., 2026; Xiang et al., 2026). Recurrence's cluster benchmark is complementary: it asks whether an agent preserves exact statements, keeps proof and disproof pressure visible, records failed routes, and selects useful next targets across related open problems.

Outside mathematics, Recurrence is closest in spirit to agents that retain reusable lessons from prior attempts. Reflexion stores verbal feedback after failures (Shinn et al., 2023), Voyager builds an ever-growing skill library from exploration and execution errors (Wang et al., 2023), and automated scientific-discovery agents organize idea generation, experimentation, and paper writing into closed loops (Lu et al., 2024). Recurrence specializes this memory pattern to mathematical research: the retained objects are conjectures, reductions, obstructions, claim maps, return maps, and next-target prompts. This is also motivated by views of mathematics as compressed structure: Mathlib dependency studies suggest that reusable definitions and lemmas help make formal space navigable (Aksenov et al., 2026; Li et al., 2026), while human-AI discovery work shows that learned systems can guide mathematical intuition (Davies et al., 2021).

Taken together, these works make the prover/formalizer layer increasingly powerful. They also make the missing human-in-the-loop layer sharper: most systems concentrate on a single theorem statement or proof object, while research mathematics often requires durable state across related questions, failed routes, examples, reusable lemmas, and judgments about what to try next. Recurrence is complementary rather than competitive with the prover layer. It targets the pre-formal and inter-run problem of stabilizing a research state: the exact target, proof routes, counterexample pressure, failed reductions, verifier objections, and next lemma that a human mathematician or later Lean agent should inspect.

4. System Overview

Recurrence extends a standard proof pipeline with a stage that runs before proof writing:

$$\begin{aligned}\text{problem} &\to \text{locked statement}\\ &\to \text{route matrix} \to \text{fresh-context critic}\\ &\to \text{proof attempt} \to \text{drift/proof verification}\\ &\to \text{research record} \to \text{memory}.\end{aligned}$$

The proof layer follows the familiar generator/checker split: a decomposer creates a proof plan, a prover writes the proof, structural and detailed verifiers attack it, and a regulator revises the proof or plan when verification fails. The recurrence layer is the main novelty.

The implementation is intentionally backend-agnostic.

Algorithm 1 Recurrence loop for one target

```
Normalize the statement: objects, hypotheses, conclusion, quantifiers.
Read human guidance and source constraints.
if mode is Explore then
    Generate positive branches: strengthenings, bridge lemmas, special cases, analogues.
    Generate negative branches: counterexample shapes, sharpness tests, missing-hypothesis tests.
else
    Build proof-method and disproof-method matrices for the locked statement.
    Ask a fresh-context critic to identify likely theorem drift and missing hypotheses.
    Test bottlenecks, edge cases, and counterexample candidates.
end if
Rank surviving branches by plausibility, checkability, source support, and target relevance.
Refuse or pause branches with unverified theorems, changed targets, or repeated fatal gaps.
Write a return map and next single-target problem.
Pass the top branch to decomposition, proving, and verification.
Extract memory: achievements, failures, frontiers, prompt lessons, and claim maps.
```

Prover, decomposer, verifier, regulator, and recurrence roles are instantiated by configurable command-line agents such as Codex, Claude, and agy (Gemini). Recurrence's novelty is not a heavily engineered formal backend, but a prompt-and-artifact protocol wrapped around these CLI agents and persistent file-based memory. This design lets mathematicians distill field-specific ways of thinking into inspectable prompts and artifacts: how to preserve hypotheses, recognize legitimate reductions, test dangerous examples, refuse false shortcuts, and decide which lemma should become the next target.

Definition 1 (Return-to-conjecture discipline). A branch is admissible only if the agent records how it affects the original target. The allowed statuses are direct progress, reduction, obstruction, finite evidence, side result, and dead end.

5. Two Modes: Explore and Attack

Explore mode. Explore mode is deliberately pre-proof. It asks the agent to behave like a mathematician walking around a problem with a notebook: formulate nearby conjectures, test special cases, search for obstructions, rank branches, and decide which ideas should not receive proof budget. This is useful when the target is open-ended, when the right formulation is unclear, or when the most valuable output may be a reduction or obstruction map rather than a final proof.

Attack mode. Attack mode is stricter. It locks the statement, enumerates proof methods, enumerates counterexample methods, tests small and edge cases, and returns one of a few operational decisions: a proof route survives, a counterexample candidate needs verification, a counterexample is verified, the target reduces to a bottleneck, the result is inconclusive, or human input is needed. This resembles the dominant behavior of many proof agents, but Recurrence treats it as one mode rather than the whole system.

Attack records are graded by outcome rather than by a solved/unsolved binary. We use O-codes for these outcomes to distinguish them from benchmark row identifiers below:

Tier	Meaning
O0	refuted by a checked counterexample or obstruction
O1	proved and independently verified
O2	conditionally proved, with named assumptions
O3	reduced to a named bottleneck lemma
O4	attacked but unresolved, with live routes recorded
O5	incoherent, drifted, or unverifiable record

The distinction matters. In research mathematics, rushing to attack can waste compute on an ill-posed target or a false strengthening. Conversely, indefinite exploration can become idea generation without progress. Recurrence separates the two behaviors and forces both to return to the conjecture.

6. Search Schemes and Artifacts

Recurrence uses several search schemes in parallel or sequence:

- Target normalization: the problem is rewritten compactly without changing meaning, with hypotheses and quantifiers made explicit.
- Positive search: bridge lemmas, special cases, strengthenings, analogues, and structural explanations.
- Negative search: obstruction tests, counterexamples, sharpness examples, missing-hypothesis checks, and false stronger claims.

- Method matrices: in attack mode, proof and disproof methods are listed with bottleneck claims and verifier formats.
- Verifier-guided pruning: proof branches are stopped when they depend on unavailable citations, change the target, or leave the same gap after revision.
- Campaign memory: extracted notes seed future prompts and prevent repeated dead ends.

The output is intentionally file-based and inspectable. A typical run writes an exploration map, conjecture ledger, approach queue, no-output decisions, approach journal, and conjecture return map. In attack mode it additionally writes an attack report, method matrix, counterexample-search log, and attack decision. After proof generation, Recurrence builds claim maps whose nodes are verifier-labeled as Yes, No, or Uncertain. These maps are not presented as formal proof certificates, but they are useful for deciding what to run next.

7. Campaign Evidence

By self-evolving we do not mean that Recurrence updates model weights. The evolution is in the mathematical state and in the prompts constructed from that state. After each run, the system extracts memory items such as useful insights, pitfall warnings, failed or revised attempts, false lemmas, ranking rationales, and next-problem recommendations. Memory is not append-only truth: later refutations demote or supersede prior items, preserving them as warnings with provenance. It also maintains cluster notes and a global claim map. A later run can therefore inherit not only a problem statement, but a history of what changed the research program.

In an internal low-dimensional topology campaign, Recurrence maintained a cluster around pseudoconvex, contact-topological, and smooth embedding questions. The memory index contained 142 low-dimensional-topology memory items, including pitfalls, useful strategies, false lemmas, verification issues, and next-problem records. Seven generated claim maps contained 51 verifier-labeled Yes nodes and 20 Uncertain nodes. These numbers are system traces rather than benchmark scores; the key question is whether the traces preserve useful research state. Table 2 gives representative examples.

Inspectable return-map example. One run studied $Y^\circ = (S^3_0(K)) \setminus \mathrm{int}(B^3) \hookrightarrow S^4$. A tempting route was to cap the boundary and apply closed-embedding obstructions to Y. Recurrence instead recorded the chain

$$Y^\circ \hookrightarrow S^4 \Rightarrow D(Y^\circ) \cong Y\#(-Y)$$
$$\Rightarrow \text{closed tools on } Y \text{ require a new cap lemma.}$$

The return map then classified the cap route as a reduction with a missing lemma, marked direct sliceness and double-sliceness claims as unsupported, refuted $\Delta_K(t) = 1$ as a necessary condition using ribbon examples, and selected the next target: compute Kawauchi local signatures of $Y\#(-Y)$ in Levine–Tristram terms. This is the intended unit of progress: a failed proof route becomes a reusable guardrail and a precise next problem.

8. Case Study: Lagrangian $\mathbb{R}P^2$ Packings

As a second case study, we considered a problem cluster around Lagrangian real projective planes in many-point blowups of $\mathbb{CP}^2$. The background is classical enough to be structured, but open-ended enough that a decisive intermediate lemma can matter for a full human–AI resolution. Shevchishin and Smirnov prove a sharp triangle-inequality criterion for when the threefold blowup of the symplectic ball contains a Lagrangian $\mathbb{R}P^2$ in the class $E_1 + E_2 + E_3$ (Shevchishin & Smirnov, 2020). Evans frames the broader nonsqueezing question for nonorientable Lagrangians: since $H^2(L;\mathbb{R}) = 0$, nonorientable Lagrangians can remain Lagrangian while the ambient symplectic form moves in an open region of cohomology (Evans, 2022). Adaloglou extends the rational-blowup perspective to Lagrangian pinwheels and studies disjunction of Lagrangian projective planes in rational surfaces (Adaloglou, 2024).

The motivating question is the many-ball analogue. Let $X_k = \mathbb{CP}^2\#k\overline{\mathbb{CP}}^2$ arise from packing k balls into $\mathbb{CP}^2$. For $k > 10$, exceptional classes become infinite and the finite triangle-inequality picture no longer gives a complete packing boundary. The research question is:

> For $k > 10$, which symplectic packing forms on X_k admit embedded Lagrangian $\mathbb{R}P^2$'s, and are such $\mathbb{R}P^2$'s smoothly unique inside a fixed $\mathbb{Z}/2$-homology class?

The desired global conclusion would describe the Lagrangian $\mathbb{R}P^2$ packing cone and classify smooth isotopy classes near its boundary. Recurrence did not, by itself, solve this global problem. It isolated a key stability lemma and supplied an argument later checked

Table 2. Representative Recurrence traces from one topology campaign.

Target	Mode	Main artifact	Lesson preserved for later runs
Brieskorn pseudoconvex finite map	explore	30-entry obstruction table; 7 Yes claim nodes	Full conjecture was too broad; finite source-backed obstruction map passed while four canonical survivors were kept unresolved.
Hyperbolic rational homology spheres with tight structures	explore	3 certified construction rows; 8 verified proof nodes	Literature search found a 2025 theorem that made negative figure-eight surgeries direct; uncertain Weeks-manifold notation was paused.
Contact invariant versus tightness	explore	Four-row implication/counterexample map; 10 Uncertain claim nodes	Global equivalence was refused; a concrete Ghiggini counterexample was retained, while one extracted row was demoted to a lemma gap.
Punctured zero-surgery embeddings	explore	Return map with 12 branches; 14 verified proof nodes	The actual punctured hypothesis gives $Y\#(-Y)$, not Y; naive closed-embedding, sliceness, and $\Delta_K = 1$ routes were recorded as dead ends.

and used in subsequent human mathematical synthesis as a decisive step toward a final resolution of the motivating problem:

> For a boundary-positive form in the $\mathbb{R}P^2$ packing framework, meaning a boundary point of the relaxed packing cone with positive margin for the relevant exceptional-class constraints, boundary non-uniqueness can be pushed to nearby honest symplectic forms: after a sufficiently small inward perturbation, one obtains non-uniqueness of smoothly embedded $\mathbb{R}P^2$'s in the same $\mathbb{Z}/2$-homology class.

The argument extracted by Recurrence is a compactness-and-continuity argument. It models the relevant obstruction inequalities through a relaxed constraint space for exceptional or near-exceptional classes, proves compactness of that constraint domain, uses continuity of the minimum exceptional-class area margin along perturbation rays, and concludes that a positive area margin at the boundary persists on a small interval of perturbations. In effect, Recurrence converted a global classification problem into a reusable stability lemma: once a boundary construction gives two smoothly non-isotopic $\mathbb{R}P^2$'s in the same $\mathbb{Z}/2$-class, the lemma supplies a checkable route for transporting that non-uniqueness into the interior of the packing cone.

This is exactly the kind of achievement our benchmark is meant to capture. The agent supplied a reusable stability lemma rather than the whole classification argument; human mathematical synthesis supplied the final resolution, to be written up as a forthcoming preprint tentatively titled Lagrangian $\mathbb{R}P^2$ Packings and Smooth Isotopy in Many-Point Blowups of $\mathbb{CP}^2$. The preprint will treat the Recurrence lemma as one component of a larger mathematical argument: first identify boundary packing classes with positive obstruction margin, then transport boundary non-uniqueness to nearby honest symplectic forms, and finally assemble the resulting examples into a statement about existence and smooth isotopy or non-isotopy of Lagrangian $\mathbb{R}P^2$'s in the $k > 10$ packing-cone picture. The partial result changes the next research step: instead of searching directly throughout the full $k > 10$ packing cone, one can focus on each boundary element, making the remaining work computable for human and machine.

Evaluation axes. For future systematic evaluation, we will measure at least five artifact-level quantities: target preservation, obstruction recovery, repeated-error reduction under campaign memory, human-rated next-target quality, and verification discipline (especially whether uncertain claims remain labeled Uncertain). These metrics do not replace theorem correctness, but they make the research-state contribution testable.

9. Cluster Benchmark: The Gompf Problem List

A Recurrence benchmark is a topic-focused cluster of related conjectures, questions, or problem-list entries. The goal is not solved-theorem count, but whether the copilot preserves exact statements, keeps proof and disproof routes visible, records failed approaches, recovers shared obstructions, and selects useful next tar-

gets across the cluster. Suitable clusters include survey problem lists, seminar open questions, conjecture families, or focused example/counterexample collections.

We focus on low-dimensional topology because it is comparatively underrepresented in standard AI-for-math benchmarks and because its problems are high-context, geometric, and category-sensitive: changing "punctured" to "closed," "Stein fillable" to "embedded Stein domain," or "locally flat" to unrestricted disks changes the mathematical problem.

We instantiate this benchmark pattern on the Gompf problem list. The input is a four-page list of open problems in low-dimensional topology: 20 numbered problems and two named conjectures, spanning smooth and symplectic 4-manifolds, handle theory, knot concordance, contact topology, Stein embeddings, homology balls, and punctured 3-manifold embeddings. The source is the 2018 Topology and Geometry of Low Dimensional Manifolds conference, a Gompf-centered meeting whose speakers worked on problems related to his mathematics; this makes the list a focused expert benchmark rather than a synthetic one, recent enough to reflect modern low-dimensional topology yet old enough that the questions have stood for years (Georgia Institute of Technology School of Mathematics, 2018). The copy used by Recurrence is included in the project repository (Baykur et al., 2018). These are not contest problems. Many are broad, research-level, and deliberately under-specified in the way community problem lists often are. This makes the list a poor benchmark for solved-theorem count, but a strong benchmark for a mathematician copilot: the agent must preserve exact statements, resist hallucinated proofs, expose bottlenecks, and produce reusable next steps. Cluster-level checks include target preservation, proof/disproof route coverage, return-to-target labels, drift detection, failure-to-guardrail conversion, verifier-labeled claim coverage, and human-rated next-target usefulness.

9.1. Protocol

The benchmark run used the full problem list as a single input file. No prior Recurrence campaign memory was imported. Because community problem lists are often under-specified, Attack mode first locks an auditable version of each statement and records the choices made. The run used explicit human guidance:

- treat the full list as the target and process entries one by one;
- do not claim that an open problem is solved without a complete, source-checkable proof;
- for each problem or conjecture, output a locked statement, proof-side route, disproof or missing-hypothesis route, bottleneck, side lemma or byproduct, failure mode, and next single-target problem file;
- if a route fails, record the failed lemma or assumption and classify it as a side lemma, obstruction, warning, dead end, or unresolved bottleneck.

The goal is therefore not "solve Gompf's list." The goal is to produce a checked research-state bundle over a cluster of open problems. This matches how we expect Recurrence to be used before formalization: as a memory- and triage-producing layer that identifies the right next mathematical statement.

9.2. Outputs

The run produced a complete artifact bundle. The recurrence stage generated an attack report, method matrix, counterexample-search log, approach queue, conjecture ledger, and conjecture return map. The proof-and-verification stage then checked the meta-claim that the bundle faithfully preserves the research state: 22 locked targets, one proof-side route and one disproof-side route for each, no false solution claims, and a recommended next single-target benchmark. The claim map contains 27 verifier-labeled Yes nodes, no No nodes, and no Uncertain nodes.

Verifier semantics. In Table 3, a Yes label means that a fresh verifier accepted a bounded artifact claim under the locked statement: the target card matches the source problem, required fields are present, proof-side and disproof-side routes return to the target, and the bundle does not promote an open problem to solved status without a complete proof. It does not mean that the verifier has proved the original open problem, produced a Lean certificate, or replaced expert mathematical review. A No label means a structural or drift claim failed; Uncertain means that a mathematical dependency, citation, or category choice could not be adjudicated from the artifact alone.

The selected next target was Gompf's second conjecture: no Brieskorn integer homology sphere, other than S^3, admits a pseudoconvex embedding in $\mathbb{C}^2$, with either orientation. Recurrence selected this target because it has an explicit countable input space, namely pairwise-coprime triples (p, q, r) and orientations; a one-object counterexample certificate, namely a nontrivial Brieskorn sphere with a Stein domain embedded in $\mathbb{C}^2$; and computable subfamily data from Seifert and plumbing descriptions. The run did not

Table 3. Gompf benchmark structural audit. The checked object is the research-state bundle, not a solution of the open problems.

Structural check	Value
Open-problem targets	22
Numbered problems	20
Named conjectures	2
Prior Recurrence memory imported	0
Locked statements produced	22
Proof-side routes produced	22
Disproof-side routes produced	22
Side lemmas/byproducts produced	22
Failure modes recorded	22
Recommended next-target files	22
Claim-map Yes/No/Uncertain	27/0/0
False open-problem solution claims	0
Selected next benchmark target	R18

prove the conjecture or find a counterexample. It reduced the next benchmark to a family-wise obstruction or a single embedding certificate.

Thus the raw counts in Table 3 should be read as structural adherence measurements, not as evidence that the recommended target is mathematically useful. The usefulness claim is evaluated separately by human ratings and by the controlled baseline comparison below.

9.3. Reusable Statements

The most important benchmark output is not a proof, but a list of reusable statements. We divide them into three types: candidate lemmas, category warnings, and next-target formulations. Table 4 gives a compact version of the 22 side lemmas and failure modes. These are intended for human evaluation: a domain mathematician can rate each item for correctness, specificity, usefulness, nontriviality, and priority.

9.4. Human Evaluation

The Gompf benchmark naturally supports a mathematician-in-the-loop evaluation. We use five axes: correctness, usefulness, specificity, nontriviality, and priority. Current artifact evidence includes zero false open-problem solution claims, 22 next-target files, 22 failure-mode records, concrete category checks, and the selected follow-up target R18. Independent expert evaluation is underway. Domain specialists will rate the 22 reusable statements on correctness, usefulness, specificity, nontriviality, and priority. The primary usefulness measurement is whether experts prefer the recommended next target and its stated rationale over baseline recommendations generated without the recurrence layer. The planned report will include per-axis means and variance, inter-rater agreement, and agreement between expert priority rankings and Recurrence's selected next target R18.

9.5. Why This Is a Benchmark

This benchmark tests properties that ordinary theorem-proving benchmarks miss. If an agent claims to have solved five open Gompf problems, it fails. If it produces only a broad literature summary, it fails. If it changes the category, it fails: punctured embeddings are not closed embeddings, abstract Stein fillability is not an embedded Stein domain in $\mathbb{C}^2$, and non-locally-flat sliceness is not locally-flat sliceness. The successful output is a set of inspectable, reusable, verifier-labeled research objects that make the next human or agent step sharper.

Controlled baseline comparison. The camera-ready ablation is designed around the reviewer-suggested question: does Recurrence choose a more useful next target than a strong non-recurrent baseline? For the same Gompf list and comparable model budget, we compare two conditions. Recurrence-On uses target locking, proof/disproof route matrices, failure ledgers, return maps, and memory extraction. Recurrence-Off uses a carefully written single prompt, followed by the same proof-and-verification stage, but does not maintain branch ledgers or campaign memory. A domain expert, blinded to condition when feasible, rates each condition's recommended next target on correctness, specificity, usefulness, nontriviality, and priority. The primary endpoint is the expert usefulness score for the selected next target; secondary endpoints are false solution claims, category drift, and presence of reusable failure modes. Pilot runs suggest that the baseline can produce plausible local summaries, but the controlled comparison is the intended measurement rather than the structural counts alone.

Reproducibility. A run takes a Markdown or LaTeX target file and writes inspectable artifacts: literature survey, recurrence map, conjecture ledger, approach queue, no-output decisions, proof/decomposition files, verification reports, token usage, and proof-effort summary. One internal run used Codex for exploration/proving and Claude Sonnet for verification. Most current runs use GPT-5.5 xhigh for exploration and proving, with verification delegated to Claude or Gemini 3.1 Pro; runtime is cluster-dependent and can exceed 10 hours. When submission infrastructure permits, we will release a public artifact bundle with representative outputs and verification reports.

10. Discussion

Recurrence's main limitation is that verification is mostly natural-language and adversarial rather than fully formal; final claims still require expert review, formalization, or independent checking. The system is therefore infrastructure for mathematician-in-the-loop research, consistent with calls for the mathematical community to stay involved in shaping AI's role (Ghosal, 2026). A natural next step is to connect claim maps to Lean: Recurrence would stabilize informal definitions and lemma boundaries, then dispatch selected claims to autoformalization and formal-verification backends. Longer term, field-sensitive self-evolution should let the agent revise its own prompt rules, verifier checks, schemas, retrieval policies, and tools from campaign failures. In geometry and topology this might add diagram-aware checks, handlebody normal forms, or obstruction ledgers; in algebraic geometry it might add scheme-theoretic normalization or deformation-theoretic warning patterns. Throughout, Uncertain should not be rewritten as success: a failed branch should become a guardrail, a missing lemma a next problem, and a byproduct a ledgered result.

Acknowledgments

The author gratefully acknowledges support from NSF LEAPS-MPS award 2609841.

References

Adaloglou, N. Embeddings and disjunction of lagrangian pinwheels via rational blow-ups, 2024. URL https://arxiv.org/abs/2405.02110.

AI for Math Workshop. 3rd AI for math workshop at ICML 2026: Toward self-evolving scientific agents. https://ai4math2026.github.io/, 2026. Accessed 2026-05-26.

Aksenov, V., Bodnia, E., Freedman, M. H., and Mulligan, M. Compression is all you need: Modeling mathematics, 2026. URL https://arxiv.org/abs/2603.20396.

Alon, N., Bloom, T. F., Gowers, W. T., Litt, D., Sawin, W., Shankar, A., Tsimerman, J., Wang, V., and Wood, M. M. Remarks on the disproof of the unit distance conjecture, 2026. URL https://arxiv.org/abs/2605.20695.

An, C., Ye, Q., Pan, M., and Zhang, J. QED: An open-source multi-agent system for generating mathematical proofs on open problems, 2026. URL https://arxiv.org/abs/2604.24021.

Barrett, C., Chaudhuri, S., Montesi, F., Grundy, J., Kohli, P., de Moura, L., Rademaker, A., and Yingchareonthawornchai, S. CSLib: The Lean computer science library, 2026. URL https://arxiv.org/abs/2602.04846.

Baykur, I., Harvey, S., Kirby, R., Li, T.-J., Lidman, T., Miller, A., Tosun, B., and Ruberman, D. Gompf problem list. https://github.com/Recurrence-Poincare/Recurrence-Agent/blob/main/gompfproblemlist.pdf, 2018. Anonymous repository copy used in the Recurrence benchmark.

Davies, A., Veličković, P., Buesing, L., Blackwell, S., Zheng, D., Tomašev, N., Tanburn, R., Battaglia, P., Blundell, C., Juhász, A., Lackenby, M., Williamson, G., Hassabis, D., and Kohli, P. Advancing mathematics by guiding human intuition with AI. Nature, 600:70–74, 2021.

Evans, J. D. A lagrangian klein bottle you can't squeeze. Journal of Fixed Point Theory and Applications, 24(47), 2022. doi: 10.1007/s11784-022-00945-w.

Feng, T., Jung, J., Kim, S.-h., Pagano, C., Gukov, S., Tsai, C.-C., Woodruff, D., Javanmard, A., Mokhtari, A., Hwang, D., Chervonyi, Y., Lee, J. N., Bingham, G., Trinh, T. H., Mirrokni, V., Le, Q. V., and Luong, T. Aletheia tackles firstproof autonomously, 2026a. URL https://arxiv.org/abs/2602.21201.

Feng, T., Trinh, T. H., Bingham, G., Hwang, D., Chervonyi, Y., Jung, J., Lee, J., Pagano, C., Kim, S.-h., Pasqualotto, F., Gukov, S., Lee, J. N., Kim, J., Hou, K., Ghiasi, G., Tay, Y., Li, Y., Kuang, C., Liu, Y., Lin, H., Liu, E. Z., Nayakanti, N., Yang, X., Cheng, H.-T., Hassabis, D., Kavukcuoglu, K., Le, Q. V., and Luong, T. Towards autonomous mathematics research, 2026b. URL https://arxiv.org/abs/2602.10177.

Gao, H.-a., Geng, J., Hua, W., Hu, M., Juan, X., Liu, H., Liu, S., Qiu, J., Qi, X., Ren, Q., Wu, Y., Wang, H., Xiao, H., Zhou, Y., Zhang, S., Zhang, J., Xiang, J., Fang, Y., Zhao, Q., Liu, D., Qian, C., Wang, Z., Hu, M., Wang, H., Wu, Q., Ji, H., and Wang, M. A survey of self-evolving agents: What, when, how, and where to evolve on the path to artificial super intelligence. Transactions on Machine Learning Research, 2026. URL https://arxiv.org/abs/2507.21046.

Georgia Institute of Technology School of Mathematics. Georgia tech @ the topology and geometry of low dimensional manifolds conference. https://math.gatech.edu/events/georgia-tech-

topology-and-geometry-low-dimensional-manifolds-conference, 2018. Conference page, accessed 2026-05-26.

Ghosal, A. "Like Space Aliens Landing": Symposium weighs effect of AI on the future of mathematics. https://stanforddaily.com/2026/05/06/future-of-mathematics-symposium-2026/, May 2026. The Stanford Daily, accessed 2026-05-25.

Gu, A., Piotrowski, B., Gloeckle, F., Yang, K., and Markosyan, A. H. ProofOptimizer: Training language models to simplify proofs without human demonstrations. In International Conference on Learning Representations, 2026. URL https://openreview.net/pdf?id=huptrb4JTa.

Harmonic. Aristotle: IMO-level automated theorem proving, 2025. URL https://arxiv.org/abs/2510.01346.

Ju, H., Gao, G., Jiang, J., Wu, B., Sun, Z., Chen, L., Wang, Y., Wang, Y., Wang, Z., He, W., Wu, P., Xiao, L., Liu, R., Dai, B., and Dong, B. Automated conjecture resolution with formal verification, 2026. URL https://arxiv.org/abs/2604.03789.

Li, X., Peng, N., Severini, S., and Shafto, P. The network structure of Mathlib, 2026. URL https://arxiv.org/abs/2604.24797.

Liu, J., Zhou, Z., Zhu, Z., Dos Santos, M., He, W., Liu, J., Wang, R., Xie, Y., Zhao, J., Wang, Q., Zhi, L., Li, J., and Li, W. Numina-Lean-Agent: An open and general agentic reasoning system for formal mathematics, 2026. URL https://arxiv.org/abs/2601.14027.

Lu, C., Lu, C., Lange, R. T., Foerster, J., Clune, J., and Ha, D. The AI scientist: Towards fully automated open-ended scientific discovery, 2024. URL https://arxiv.org/abs/2408.06292.

OpenAI. An OpenAI model has disproved a central conjecture in discrete geometry. https://openai.com/index/model-disproves-discrete-geometry-conjecture/, May 2026. Accessed 2026-05-25.

Poincare, H. Intuition and logic in mathematics. https://mathshistory.st-andrews.ac.uk/Extras/Poincare_Intuition/, 1905. English translation hosted by MacTutor, accessed 2026-05-25.

Ravi, N., Ying, K., Nesterov, V., Krishnan, R., Uskuplu, E., Xia, B., Aswedige, J., and Nashold, L. Formalproofbench: Can models write graduate level math proofs that are formally verified?, 2026. URL https://arxiv.org/abs/2603.26996.

Ren, Z. Z., Shao, Z., Song, J., Xin, H., Wang, H., Zhao, W., Zhang, L., Fu, Z., Zhu, Q., Yang, D., et al. DeepSeek-Prover-V2: Advancing formal mathematical reasoning via reinforcement learning for subgoal decomposition, 2025. URL https://arxiv.org/abs/2504.21801.

Shao, Z., Luo, Y., Lu, C., Ren, Z. Z., Hu, J., Ye, T., Gou, Z., Ma, S., and Zhang, X. DeepSeekMath-V2: Towards self-verifiable mathematical reasoning, 2025. URL https://arxiv.org/abs/2511.22570.

Shevchishin, V. and Smirnov, G. Symplectic triangle inequality. Proceedings of the American Mathematical Society, 148(4):1389–1397, 2020. doi: 10.1090/proc/14842.

Shinn, N., Cassano, F., Labash, B., Gopinath, A., Narasimhan, K., and Yao, S. Reflexion: Language agents with verbal reinforcement learning. In Advances in Neural Information Processing Systems, 2023.

Stanford University. Future of mathematics symposium. https://events.stanford.edu/event/future-of-mathematics-symposium, 2026. Accessed 2026-05-25.

Tao, T. There's more to mathematics than rigour and proofs. https://terrytao.wordpress.com/career-advice/theres-more-to-mathematics-than-rigour-and-proofs/, 2007. Accessed 2026-05-25.

Trinh, T. H., Wu, Y., Le, Q. V., He, H., and Luong, T. Solving olympiad geometry without human demonstrations. Nature, 625:476–482, 2024.

Wang, G., Xie, Y., Jiang, Y., Mandlekar, A., Xiao, C., Zhu, Y., Fan, L., and Anandkumar, A. Voyager: An open-ended embodied agent with large language models, 2023. URL https://arxiv.org/abs/2305.16291.

Wang, H., Unsal, M., Lin, X., Baksys, M., Liu, J., Dos Santos, M., Sung, F., Vinyes, M., Ying, Z., Zhu, Z., et al. Kimina-Prover Preview: Towards large formal reasoning models with reinforcement learning, 2025. URL https://arxiv.org/abs/2504.11354.

Xiang, K., Zhang, T. J., Liu, Z., Zhou, B., Tang, Y., Yu, J., Lu, J., Huang, S., Li, H., Zhang, L., et al. SeePhys Pro: Diagnosing modality transfer and blind-training effects in multimodal RLVR for physics reasoning, 2026. URL https://arxiv.org/abs/2605.09266.

Xin, H., Guo, D., Shao, Z., Ren, Z., Zhu, Q., Liu, B., Ruan, C., Li, W., and Liang, X. DeepSeek-Prover: Advancing theorem proving in LLMs through large-scale synthetic data, 2024a. URL https://arxiv.org/abs/2405.14333.

Xin, H., Ren, Z. Z., Song, J., Shao, Z., Zhao, W., Wang, H., et al. DeepSeek-Prover-V1.5: Harnessing proof assistant feedback for reinforcement learning and Monte-Carlo tree search, 2024b. URL https://arxiv.org/abs/2408.08152.

Table 4. Reusable statements extracted from the Gompf benchmark. Each row is a candidate research-state object, not a claimed theorem.

ID	Reusable statement	Failure mode or warning
R1	Audit $b^+ = 1$ symplectic/Seiberg–Witten chamber data for a homology $\mathbb{CP}^2$.	Reduces to the unknown smooth uniqueness problem for manifolds homeomorphic to $\mathbb{CP}^2$.
R2	Split torsion-canonical symplectic 4-manifolds by π_1, χ, signature, and torsion order before asking for diffeomorphism type.	Homological classification alone cannot determine diffeomorphism type in dimension 4.
R3	Choose a specific Heegaard Floer invariant package and test whether it vanishes on T_{n+1} while detecting a certified T_n witness.	The invariant may only detect sliceness or the wrong filtration level.
R4	Certify 2-torsion in T_n/T_{n+1} by proving $K \in T_n$, $K \notin T_{n+1}$, and $2K \in T_{n+1}$.	A candidate is not useful unless membership, nonmembership, and doubling are all certified in the quotient.
R5	Distinguish infinite rank from a split $\mathbb{Z}^\infty$ summand by requiring coordinate homomorphisms or a retraction.	Infinite independent classes do not by themselves prove a direct summand.
R6	A Whitehead-double counterexample certificate needs a smooth slice disk for $\mathrm{Wh}(K)$ and a nonsliceness certificate for K.	Many classical invariants are killed by Whitehead doubling.
R7	Make a move-by-move Cerf compatibility table for lifting Gay's oriented proof to spin or characteristic bordism.	One incompatible local move would obstruct the direct proof route.
R8	Treat $(b_1, b_2) = (2, 1)$, giving $\chi = -1$, as the first small negative-Euler threat profile.	Betti arithmetic does not construct a symplectic non-ruled manifold.
R9	Formalize the exact symplectic BMY inequality and hypotheses before proof or counterexample search.	Without a precise inequality, the target is not verifier-ready.
R10	Analyze positive-definite BMY through handle/perfect-Morse control, especially its interaction with R11.	The route may reduce exactly to the unsolved perfect-Morse problem.
R11	Separate algebraic simplification of presentations from geometric handle cancellation.	Algebraic simple connectivity does not imply handle cancellation.
R12	Build a dual-handlebody test for necessary 3-handles in weight-one homology-sphere constructions.	The route may only recover already-known 1-handle obstructions.
R13	Test associated-manifold packages using nonconcordant knots with equivalent associated manifolds but different concordance behavior.	Yasui-type examples warn that associated manifolds can forget concordance information.
R14	Define "interesting" for satellite-induced homomorphisms before evaluating a theorem.	Otherwise only identity or constant maps may qualify.
R15	Combine an explicit hyperbolic rational homology sphere with an exhaustive contact/tightness classification.	Constructing tight structures on some families cannot rule out all counterexamples.
R16	Compare Mark–Tosun nonvanishing with an independent tightness criterion in the exact negative-stabilized surgery family.	A tight example with zero invariant in the exact family would kill the route.
R17	A Brieskorn embedding certificate must include a nontrivial Brieskorn sphere, orientation, Stein domain, and actual embedding in $\mathbb{C}^2$.	Abstract Stein fillability is not enough.
R18	Decompose Brieskorn integer homology spheres by triples (p, q, r) and orientation, then prove a pseudoconvex embedding obstruction family by family.	A parametric obstruction leaving infinitely many triples uncovered does not prove the conjecture.
R19	Fix the disk category carefully because non-locally-flat disks are allowed unless a variant says otherwise.	Standard sliceness obstructions often require local flatness.
R20	Separate free actions from nonfree actions and audit fixed-point local models for equivariant fillings.	A proof handling only free actions does not address the full problem.
R21	Choose an explicit surgery-theoretic topological 4-manifold before computing smoothing obstructions.	Without a concrete model, there is no obstruction to compute.
R22	Derive concordance-invariant restrictions from $S^3_0(K) \setminus B^3 \hookrightarrow S^4$, not from a closed embedding of $S^3_0(K)$.	Closed-embedding arguments do not automatically apply to punctured embeddings.