CONTINUAL GRAPH MEMORY FOR MATHEMATICAL RESEARCH AGENTS

Anonymous authors
Paper under double-blind review

ABSTRACT

Using frontier agent harnesses to tackle mathematical research problems has emerged as an effective means of advancing mathematics. However, solving frontier problems in mathematics may require a massive number of agents working in parallel for extended periods to construct proofs, thereby generating an enormous volume of intermediate proof results. Organizing these intermediate results throughout a long-horizon proof-search process and reusing knowledge gained from prior explorations remain major challenges. We present Ansatz, a mathematical research agent built around CONTINUAL GRAPH MEMORY, a graph-based, evolvable, cross-problem mathematical research memory system that explicitly organizes the entire proof search process and reuses information from exploration trajectories of previous problems. Specifically, we develop a unified graph memory that represents all intermediate exploration results, including facts, plans, and counterexamples, together with edges that explicitly represent the relationships among them; dependency-aware retrieval supplies precisely targeted local context; an evidence-sensitive curator updates the research frontier and distills lessons from prior attempts; and scoped recall surfaces earlier statements and negative findings for local re-proving rather than uncritical reuse. Experiments cover runs across all ten *First Proof Second Batch* problems, together with four component studies. Ansatz reports closure on all ten research tasks, demonstrating its ability to sustain and resume long-horizon mathematical search. Beyond these problems, Ansatz also produces solutions to the Jamison caterpillar conjecture and Erdős Problems 289, 348, and 488 without human intervention, and makes partial progress on several open problems, illustrating its strong ability to solve open mathematical research problems. Our code and results are available at https://anonymous.4open.science/r/continual-graph-memory-B671/

1 INTRODUCTION

Mathematical research is not a sequence of isolated proofs. Difficult problems generate partial reductions, failed constructions, useful counterexamples, promising analogies, and lemmas whose significance may become clear only much later. Language-model agents can already explore multiple lines of reasoning through stepwise inference and sampled solution paths (Wei et al., 2022; Wang et al., 2023b), while research-oriented systems such as DANUS and Albilich extend this process with parallel workers, persistent proof state, and verification (Liu et al., 2026; Gong et al., 2026). But persistence alone is not enough. A research agent must decide what to remember, retrieve, trust, and reuse. These memories serve different roles: a method suggests what to try, a lemma is valid only when its assumptions hold, and a plan records a search choice rather than a proven fact. Treating them all as the same kind of text hides their evidential status and reuse conditions. This is especially risky for LLMs, which may store a flawed intermediate argument and later propagate the error through persistent memory.

We introduce **Ansatz**, a mathematical research agent built around CONTINUAL GRAPH MEMORY, an evolvable memory architecture designed to preserve these distinctions. CONTINUAL GRAPH MEMORY combines three complementary forms of memory: a project-local dependency graph of accepted proof records, a typed exploration graph connecting facts, plans, and counterexamples, and many other intermediate findings. An evidence-sensitive curator updates the exploration frontier and records reusable experience, while a separate verifier controls the admission of ordinary mathemat-

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

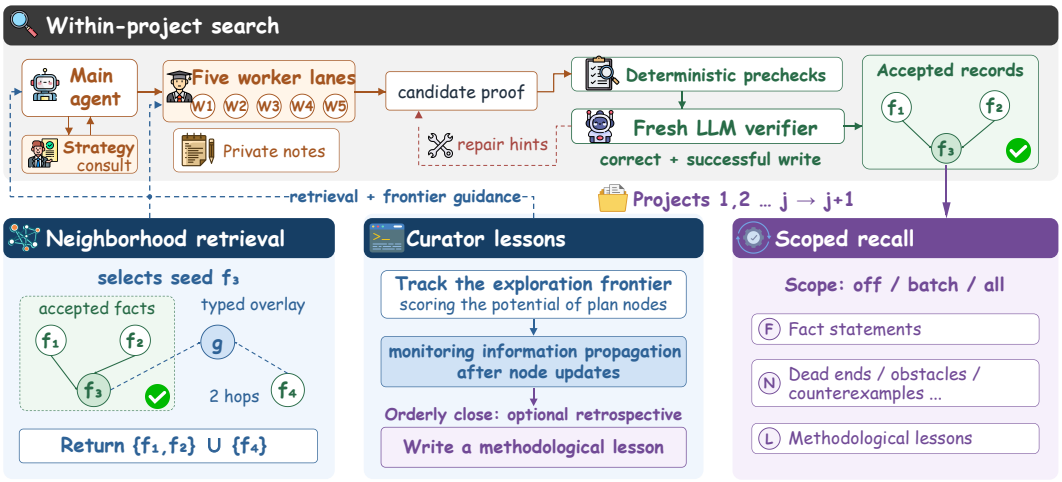

Figure 1: **Continual Graph Memory.** (a) The harness consists of workers, strategy consultation, and verifiers. The graph memory system coordinates the exploration process of all agents. Agents update and access the memory through tool calls. (b) Dependency-aware retrieval leverages the dependency information in the graph to provide multi-hop information about the target theorem. (c) Curator organizes the whole graph, update node states based on edge-based information propagation, revisits stale annotations and record structured lessons at orderly project close. (d) Scoped recall retrieves fact statements, negative findings, and methodological lessons together with provenance. The twenty most recent eligible lessons are included in the context of a later curator. Cross-project statements must be re-admitted locally under the agent protocol, while temporal isolation requires sequential orchestration.

ical facts. Dependency-aware neighborhood retrieval supplies only locally relevant context, and scoped recall exposes potentially useful material from eligible prior projects for local re-proving rather than unconditional reuse. Figure 1 summarizes this interaction.

We evaluate Ansatz on all ten problems from *First Proof Second Batch* (Abouzaid et al., 2026). Ansatz reports local closure on all ten tasks. We compare these outcomes with those of Danus and the four expert-reviewed baselines reported in the benchmark publication. Ansatz reports local closure on more tasks than any of these comparison methods. Because these factors vary simultaneously, the additional closures cannot be attributed to memory alone. We therefore complement task-level outcomes with matched resource and memory diagnostics from the earlier configuration, four component studies, and post-hoc review of accepted records to study how memory is allocated, retrieved, and occasionally misused.

Beyond *First Proof Second Batch* (Abouzaid et al., 2026), we further evaluate Ansatz in open-ended mathematical research settings on a collection of difficult open problems. Ansatz produces complete solutions to four problems, including the Jamison caterpillar conjecture and Erdős Problems 289, 348, and 488, without human intervention, and makes meaningful progress toward the targets on six additional problems. These experiments test whether Ansatz can sustain long-horizon mathematical search beyond benchmark-style tasks. Taken together, the results suggest that the system is capable not only of partial progress on challenging open problems, but also of independently reaching complete solutions.

Our contributions are: **(1) An graph-based, evolvable and cross-problem memory architecture for mathematical research.** We introduce a structured memory system that separates certified proof records, exploratory state, and cross-project experience, combining dependency-aware local retrieval, evidence-sensitive curation, and scoped recall under explicit admission boundaries. **(2) An auditable empirical study of persistent mathematical research.** We report complete task-level results on *First Proof Second Batch*, comparisons with published expert-reviewed baselines, matched resource and memory diagnostics, four component studies, and open-ended research cases study together with explicit verification qualifications and the full proof artifact. **(3) Real Advances in Mathematics.** We allocate resources to run Ansatz on dozens of open problems in mathematics and have successfully tackled or partially solved several problems at the frontier of mathematics, leading to genuine advances in the field.

2 Related Work

Proof search and orchestration. Language-agent systems interleave reasoning with tool use (Yao et al., 2023b) and explore branching or graph-structured reasoning trajectories (Yao et al., 2023a; Besta et al., 2024). For mathematical research, Rethlas and Archon combine informal conjecture resolution with a separate formalization stage (Ju et al., 2026). Danus introduces the immediate architectural foundation for our work: parallel workers, a stateless verifier, and a shared graph of accepted facts with explicit proof dependencies (Liu et al., 2026). Albilich likewise organizes long-horizon research around persistent proof state, verification, human steering, and computer algebra (Gong et al., 2026). Our work builds a unified graph memory that targets the proof-search process for mathematical research problems, represents the detailed relationships among different types of intermediate proving information rather than facts alone, and enables cross-problem learning and reuse of the graph memory.

Evolving agent memory. Reflexion stores linguistic feedback from prior attempts (Shinn et al., 2023), Voyager accumulates reusable executable skills (Wang et al., 2023a), and MemGPT manages information across memory tiers (Packer et al., 2023). More recent systems make memory organization itself adaptive: A-MEM links and evolves structured notes (Xu et al., 2025), Mem0 extracts and consolidates conversational memories, including a graph-based variant (Chhikara et al., 2025), and G-Memory connects insights, queries, and multi-agent interaction histories (Zhang et al., 2025). ISM is especially close in its use of an actively maintained strategy bank for mathematical reasoning with a frozen model and symbolic verification (Dixit & Oates, 2026). CoEvoSkills couples evolving skill packages with an independent surrogate verifier that supplies revision feedback (Zhang et al., 2026). Our work systematically studies evolving memory for mathematical research tasks and integrates it into a mathematical research system.

3 Continual Graph Memory

Continual Graph Memory is the memory layer of Ansatz. It fixes what an agent may store, who may write to each store, how stored material is retrieved, and what carries over to the next problem. This section gives the intuition; exact definitions, defaults, and algorithms are in Appendix A.

3.1 Overview

Ansatz works through a sequence of research problems. Each problem is a project with a task statement and a compute budget. Within a project, many worker sessions start and end, so progress must live outside any single context window. Across projects, experience should transfer, but a statement proved in an earlier project must not count as proved in the current one.

Several LLM roles act on shared memory through tools. *Workers* explore and write proofs. A *verifier* reviews each submitted proof in a fresh session. A *curator* reads the exploration state and scores which routes look promising. A *main agent* assigns workers to routes, requests strategy consultations, and decides when to stop.

3.2 Admitting facts

A worker submits a candidate through `fact_submit`: a statement, a proof, and the identifiers of the accepted facts the proof relies on. The candidate becomes a fact only if three steps all succeed. First, *deterministic prechecks* reject vacuous content, known unsupported proof patterns, and any cited predecessor that has been revoked. Second, a *fresh verifier session* reads the statement and proof and returns `correct`, or `wrong` with repair hints. Third, the fact store *writes* the record and returns its identifier. Each fact gets a content-addressed identifier, so resubmitting the same statement and proof yields the same record. One dependency edge is recorded from each cited predecessor. Because this is the only path into the fact graph, curator scores, discussion posts, and recalled statements can never become accepted facts on their own.

If an accepted fact is later found faulty, `fact_revoke` removes it together with every fact that declared it as a dependency, and logs the decision. Throughout the paper, "accepted" means accepted by an LLM verifier, not checked by a proof assistant.

A Worker: propose and repair

Context `<target>`, `<assignment>`, `<guidance>`; current project stores.

Read the lane and current evidence. Publish findings, including failed approaches, with explicit support and typed links. Submit a self-contained proof with local predecessor IDs. Repair reported errors and gaps.

Submit `fact_submit(<statement>, <proof>, <predecessors>)`

Reuse Cite the returned local `fact_id` only after admission and a successful write.

B Verifier: inspect the proof

Input `<run_id>`, `<statement>`, `<proof>`; verifier contract.

Check each proof item in order, including hypotheses, citations, and theorem applications. Record every critical error and gap. Return `correct` only when both lists are empty; otherwise return `wrong` with repair hints.

Output `verification_report`, `verdict`, `repair_hints`.

Invocation A fresh session starts only after prechecks pass; it does not write facts.

C Curator: score with evidence

Context `<frontier>`, `<evidence_ids>`, `<guidance>`, eligible lessons.

Read full evidence before judging. Apply the contract's AND–OR rules; revisit stale nodes. Leave unlinked plans unscored. Rank promise with an exploration bonus and cite the evidence for each judgment.

Annotate Node, status, score in $[0, 1]$, `evidence_considered`, comment.

At close Optionally record a methodological `pattern`, `evidence`, and `adjustment`.

D Recall: transfer under scope

Query `<query>`; channels `lessons`, `methodology`, `facts`.

Respect the operator's `off`/`batch`/`all` setting. If disabled, continue with current stores. Inspect source provenance. Treat retrieved statements as leads: reprove and submit locally before using them as predecessors.

Return Channel-specific ranked records with `<source_project>` and provenance.

Boundary A shared batch label alone does not enforce chronological visibility.

Figure 2: **Condensed contract templates.** The four panels expose the inputs, instructions, and output interfaces of the implemented memory loop. These are editorial condensations of the supplied worker/verifier/curator contracts and recall tool documentation, not verbatim runtime prompts. Acceptance is an LLM judgment; curator rules and methodological lesson restrictions are agent instructions.

3.3 Organizing Exploration

Typed exploration graph. Most of a research run consists of attempts rather than proofs. Workers record attempts as nodes such as plans, findings, goals, counterexamples, and obstacles, and connect them to each other and to facts with `graph_link`, choosing from fourteen relation types such as `decomposes-into`, `refutes`, and `promoted-to`. A dependency edge in the fact graph says "this proof uses that fact"; an exploration edge only says "these two are related in this way."

Curator annotations. Through `graph_annotate`, the curator attaches to a node a status (such as open, active, closed, dead, or blocked), an optional score in $[0, 1]$, the evidence it read, and a short rationale. Annotations are appended, and the latest one is the node's current view. The curator's contract says when a route may be marked closed: a decomposition closes its parent only when every obligation, including any reassembly step, is closed; a reduction needs an accepted bridge; an equivalence needs both directions. These rules are prompt instructions; the store only checks relation types and score ranges (Appendix A.3).

Frontier and staleness. `graph_frontier` returns a ranked worklist of open routes: linked nodes before unlinked, scored before unscored, higher scores first. If a node gains a new edge after

Table 1: Experimental results for First Proof Second Batch. ✓: Pass (essentially flawless or minor revisions); △: major revisions; ×: reject; —: no submission. ‡ ETH additionally calls GPT-5.5, Gemini 3.1 Pro Preview, Claude Opus 4.7.

	Official Participating Team				Our Reported results			
Task	ETH	UCLA	OpenAI	Princeton	Plain	Codex	Danus	**Ansatz**
LLM	GPT-5.5 Pro‡	GPT-5.5 Pro	GPT-5.5 Pro	Gemini 3.1 Pro Preview	GPT-5.6 Sol	GPT-5.6 Sol	GPT-5.6 Sol	GPT-5.6 Sol
01 Automorphism	✓	✓	✓	×	✓	✓	✓	✓
02 Möbius	✓	✓	✓	×	✓	✓	✓	✓
03 Bernoulli	✓	×	×	×	×	×	×	✓
04 Dilation	×	×	×	×	×	×	×	✓
05 SPDE	✓	×	×	×	×	✓	✓	✓
06 Tree-lattice	—	✓	✓	×	×	✓	✓	✓
07 Matching	✓	✓	✓	✓	✓	✓	✓	✓
08 Dressian	×	△	×	×	×	×	✓	✓
09 Hilbert	✓	✓	✓	×	✓	✓	✓	✓
10 von Neumann	△	×	△	×	×	×	×	✓
Overall	6/10	5/10	5/10	1/10	4/10	6/10	7/10	**10/10**

its last annotation, that annotation is flagged *stale* and the curator revisits it. Judgments therefore stay tied to the evidence that existed when they were made.

3.4 Retrieving context

A fresh worker session cannot read every earlier interaction, so it retrieves (Appendix A.4). `fact_search` ranks stored statements with BM25 (Robertson & Zaragoza, 2009) and returns matching identifiers and statements. `fact_neighbors` then starts from one fact and returns its declared predecessors, optionally its direct dependents, and the accepted facts that share an exploration edge with it or sit two edges away through a non-fact node, together with the connecting paths and edge labels. In Figure 1b, a query for f_3 returns its predecessors f_1, f_2 and also f_4, which shares no words with f_3 but supports the same goal g. The default budget is 25 statements, predecessors first. Both tools return statements, not proofs; the worker decides what to read and what to reuse.

3.5 Carrying experience across projects

Scope. An operator setting decides which other projects are visible: `off` (none, the default), `batch` (projects sharing the current batch label), or `all`. Scope filters by membership, not by time. An evaluation in which only earlier projects should be visible must enforce that ordering outside the memory system (Appendix A.5).

Recall. `memory_recall` searches three channels separately with BM25—fact statements, negative findings (dead ends, obstacles, counterexamples), and lessons—and returns five results per channel by default, each tagged with its source project. A recalled fact arrives as a statement without a proof and has no standing in the current project. A worker who wants to use it must re-prove it and submit it through `fact_submit` like any other candidate. Negative findings help workers avoid repeating known failures.

Lessons. When a project closes in an orderly way, the curator may call `lesson_add` with a pattern it noticed in its own scoring, the outcome evidence, and a proposed adjustment. The next project's curator starts with the twenty most recent eligible lessons in its context; workers and the main agent query lessons on demand. Adaptation across projects thus happens in inspectable text rather than in model weights.

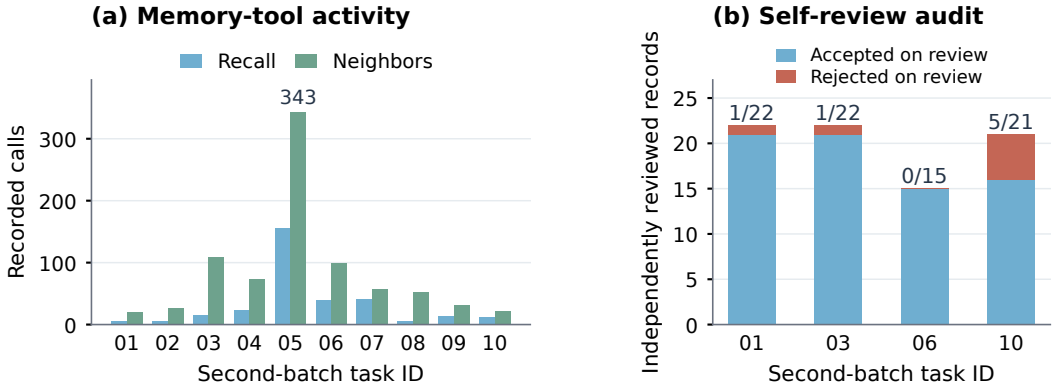

Figure 3: (a) Memory retrieval activity across First Proof Second Batch tasks. The bars report the number of memory retrieval calls made by Ansatz for each task. Blue denotes calls to the embedding-based `memory_recall` tool, while green denotes calls to the graph-based multi-hop `fact_neighbors` tool. (b) Independent verifier review of selected records approved by workers through self-review. Bar labels indicate rejected/reviewed counts. We use a census for Tasks 01 and 06 and random sampling for Tasks 03 and 10.

4 Experiments

Our experiments consist of two parts. In Section 4.1, we evaluate Ansatz on the First Proof Second Batch benchmark. Since this is a research-level mathematics benchmark for which solutions are publicly available, we disable Ansatz's web search functionality to prevent access to existing solutions. In Section 4.2, we evaluate Ansatz on genuinely open mathematical problems and report several results obtained by Ansatz on these problems.

4.1 First Proof Benchmark

4.1.1 Experimental setup

We use all ten problems from the *First Proof Second Batch* (Abouzaid et al., 2026) as our evaluation set. Since First Proof Second Batch is a research-level mathematics benchmark whose solutions have already been made public, we disable the web search functionality of the agent systems during our new evaluations. We report results for Ansatz, Danus, Codex, and plain GPT-5.6 Sol, all using GPT-5.6 Sol as the base model. We compare these results against those of the four teams included in the official First Proof Second Batch report.

4.1.2 Performance of Ansatz

Experimental results for First Proof Second Batch are shown in Table 1. Ansatz substantially outperforms both the official participating teams and the other GPT-5.6-based baselines on the First Proof Second Batch benchmark. Ansatz solves all 10/10 tasks, achieving the highest overall pass rate among all evaluated systems, compared with 7/10 for Danus, 6/10 for Codex, and 4/10 for the plain GPT-5.6 solution, as well as 6/10, 5/10, 5/10, and 1/10 for the four official participating teams. The gains are particularly pronounced on problems for which most other systems fail. In particular, Ansatz is the only system in the table to pass Task 04 (Dilation) and the only system to pass Task 10 (von Neumann): all other systems are rejected on Dilation, while the best official submissions on von Neumann require major revisions. Ansatz also passes Task 03 (Bernoulli), where all systems except the ETH submission fail, as well as Tasks 05 (SPDE), 06 (Tree-lattice), and 08 (Dressian), on which multiple competing systems fail or require major revisions. These results suggest that the gains from Ansatz cannot be explained by the GPT-5.6 backbone alone. With the same GPT-5.6 Sol backbone, Ansatz improves over the plain baseline from 4/10 to 10/10, while also outperforming Codex (6/10) and Danus (7/10), indicating that the proposed system design substantially improves performance across a diverse set of research-level mathematical problems.

Table 2: Ablation study of the components in Ansatz. Since conducting ablation studies of Ansatz on research-level tasks is very costly, we selected seven problems for ablation studies A and B, and four problems for ablation studies C and D.

Component study	01	02	03	05	06	07	08	09	10	Closed
Ansatz	✓	✓	✓	✓	✓	✓	✓	✓	✓	10/10
A. Flat fact notebook	✓	✓	—	✓	✓	✓	✓	✓	—	7/7
B. No exploration graph or curator	✓	✓	—	×	✓	✓	✓	✓	—	6/7
C. Worker self-review	✓	—	×	—	✓	—	—	—	×	2/4
D. No strategy consult	✓	—	×	—	×	—	—	—	×	1/4

4.1.3 Memory activity and evidence reuse

The memory retrieval activity across First Proof Second Batch tasks is shown in Figure 3(a). Memory activity varies substantially across tasks, while graph-based retrieval consistently accounts for the majority of memory accesses. `fact_neighbors` is invoked more frequently than the embedding-based `memory_recall` tool on every Second Batch problem, indicating that the agent frequently expands retrieved facts through multi-hop graph traversal rather than relying solely on embedding-based similarity search. The difference is particularly pronounced on Task 05 (SPDE), where the system makes 343 `fact_neighbors` calls, together with a large number of `memory_recall` calls, indicating substantially more intensive memory exploration than on the other tasks. Tasks 03, 04, and 06 also trigger relatively high levels of graph traversal, whereas Tasks 01, 02, and 10 involve comparatively little memory interaction. The substantial variation across tasks suggests that Ansatz does not operate with a fixed retrieval budget, but instead adjusts its memory activity to the course of reasoning on each problem. Notably, Ansatz successfully solves all ten tasks despite large differences in retrieval volume: for example, Task 05 requires intensive memory interaction, whereas Task 10 is solved with relatively few retrieval calls. Thus, retrieval frequency alone is not indicative of end-task success; rather, the amount of memory exploration required can vary considerably across problems.

4.1.4 Component Ablation

The ablation study of the components in Ansatz is shown in Table 2. For the seven tasks used in ablations A and B, the full Ansatz system succeeds on all seven. Replacing the fact graph with a flat fact notebook (A), while preserving the stored facts but removing graph edges, retains the same 7/7 success rate. Thus, on this subset, the explicit graph structure of the fact memory does not yield an observable improvement in end-task performance. In contrast, removing the exploration graph and curator (B) introduces a failure on Task 05, reducing performance from 7/7 to 6/7 and providing evidence that the exploration-and-curation mechanism contributes to successful problem solving.

For ablations C and D, we evaluate four tasks (01, 03, 06, and 10), all of which are solved by the full Ansatz system. Replacing the independent verifier with worker self-review (C) introduces failures on Tasks 03 and 10, reducing performance from 4/4 to 2/4. Removing strategy consultation (D) has an even larger effect, introducing failures on Tasks 03, 06, and 10 and leaving only Task 01 solved, for an overall success rate of 1/4.

Figure 3(b) reports an independent re-review of records that had previously been accepted by workers through self-review. The audit further reveals a weakness of relying on workers to verify their own outputs: among 80 records accepted by worker self-review, 7 are subsequently rejected by an independent reviewer. The rejected records are concentrated most heavily on Task 10, where 5 of 21 accepted records fail independent review; one rejected record is also found on each of Tasks 01 and 03, while none are found among the 15 reviewed records for Task 06. Notably, the self-review ablation fails on Tasks 03 and 10, both of which contain records that were accepted by worker self-review but rejected upon independent re-review. This correspondence does not by itself establish that these records caused the end-task failures, but it provides direct evidence that worker self-review can admit erroneous intermediate facts. Task 10 is particularly informative, as it exhibits both the highest rate of independently rejected records and an end-task failure under the self-review ablation.

Table 3: Ansatz outcomes on open problems. Solved indicates that the target is met, while Partial indicates meaningful progress toward the target without fully meeting it.

Problem	Current best	Ansatz	Target	Solved
Jamison caterpillar conjecture (Jamison, 1983; 1984)	$n \leq 24$ (Mol & Oellermann, 2019)	all n	all n	Solved
#289 Erdős–Graham interval problem (Erdős & Graham, 1980)	Open (Bloom, 2026b)	all large k	all large k	Solved
#488 density of multiples (Bloom, 2026e)	Open (Bloom, 2026e)	counterexample; ratio $> 256/67$	decide factor-2 bound	Solved
#348 complete-sequence deletion problem (Erdős & Graham, 1980)	$m = 0, 1$ known (Bloom, 2026d)	exactly $m = 0, 1$	classify all $m < n$	Solved
#811 balanced-rainbow problem (Erdős & Tuza, 1993)	most q (Axenovich & Clemen, 2024); K_4 (Clemen & Wagner, 2023)	cliques $q \geq 4$	all graphs	Partial
#644 pair-transversal conjecture (Erdős et al., 1992)	$7/8$ (Fon-Der-Flaass et al., 1999)	$6/7$	$3/4$	Partial
#561 star-forest Ramsey conjecture (Burr et al., 1978)	one star; equal pair; all odd (Davoodi et al., 2025)	two stars, any forest	all lists	Partial
#993 unimodality of independence sequences for tree (Bloom, 2026i)	$n/8$ (Basit & Galvin, 2021)	$\frac{131}{520}n$	up to the peak ($\leq 0.276\,n$)	Partial
#1063 binomial divisibility (Erdős & Selfridge, 1983)	subexponential upper bound claimed (Erdős Problem a Day, 2026)	super-polynomial lower; refined upper	sharp growth of n_k	Partial
#302 reciprocal-sum-free sets (Erdős & Graham, 1980)	lower $> 5/8$; upper < 0.860853 (preprint) (Khanukov, 2026)	$f(N)/N \to c$; explicit lower gain	determine c	Partial

Overall, these results provide evidence for the practical value of the exploration-and-curation mechanism, independent verification, and strategy consultation. In contrast, we do not observe a measurable benefit from the fact-graph structure itself in this ablation, suggesting that, for the tasks studied here, how facts are explored, validated, and used strategically may matter more than whether they are stored with explicit graph edges.

4.2 Open Problems

Here we run the same system on open problems. Table 3 records the published current best, the Ansatz claim, the conjectured target, and whether that claim solves the problem. Ansatz produces complete solutions to four mathematical problems: the Jamison caterpillar conjecture and Erdős Problems 289, 488, and 348, without human intervention. Ansatz also makes meaningful progress toward the targets on six additional problems. The statements of the four solved problems are shown below, while the statements of the other problems are provided in Appendix B.

Jamison. The mean subtree order of a tree is the average order of its nonempty connected induced subgraphs. Jamison conjectured that every n-vertex maximizer is a caterpillar (Jamison, 1983; 1984). Mol and Oellermann verified this through $n = 24$ and restricted the limbs of any maximizer (Mol & Oellermann, 2019). Ansatz proves that every n-vertex maximizer is a caterpillar: a shortest terminal arm is reduced to a broom, then straightened by exact polynomial certificates. This fully solves the conjecture for every n.

289. Erdős and Graham asked whether 1 can be written as a sum of reciprocals over k finite intervals of positive integers, for every sufficiently large k (Erdős & Graham, 1980; Bloom, 2026b). The catalogue form requires the intervals to be pairwise disjoint and non-adjacent, each of length at least two; that separated form was open. Ansatz produced a proof that such a threshold k_0 exists. The argument uses a Pell anchor, equal-mass packets, and a summable reserve; the threshold is not effective, because the last step uses compactness. The unrestricted variant, which allows overlap, already has a short affirmative argument (Kovač, 2025).

488. For a finite nonempty set A of positive integers, let B be the set of their multiples and write $d_A(t) = |B \cap [1, t]|/t$. Erdős asked whether $d_A(m) < 2d_A(n)$ whenever $m > n \geq \max A$ (Bloom, 2026e). Ansatz constructs a finite A and admissible m, n with $d_A(m)/d_A(n) > 256/67 > 2$, disproving the inequality. Its elements have prime factors in a fixed finite set and are at least a threshold, but division by any prime factor puts them below it. Their multiples have low density near $\max A$, while an exact residue-class count gives a higher density at a later cutoff.

348. A sequence is complete if every sufficiently large positive integer is a sum of terms at distinct indices; repeated values are allowed. The question asks which $0 \leq m < n$ admit a sequence that stays complete after every m-term deletion but becomes incomplete after every n-term deletion (Erdős & Graham, 1980; Bloom, 2026d). The cases $m = 0, 1$ were known; van Doorn excluded $m \geq 2$ under the stronger definition requiring every positive integer to be represented (Bloom, 2026d). Ansatz proves that, under eventual completeness, the admissible pairs are exactly $(0, n)$ for $n \geq 1$ and $(1, n)$ for $n \geq 2$. Powers of two and the Fibonacci sequence supply the two families. For the converse, Ansatz proves that any sequence surviving every two-term deletion admits a deletion of any prescribed finite size that preserves completeness, ruling out $m \geq 2$.

For the Jamison caterpillar conjecture, our result is, to the best of our knowledge, the first complete resolution of the conjecture. For the other three solved problems: Erdős–Graham Problem 289, Problem 488 on the density of multiples, and Problem 348 on complete-sequence deletion, we found, after the Ansatz runs had been completed, that recent proofs had also been obtained by other groups. However, a post hoc audit of the complete retrieval and reasoning traces showed that Ansatz did not retrieve, access, or rely on these contemporaneous proofs while solving the problems. Its solutions were derived independently from the information available during the runs. Moreover, the resulting proofs use approaches that are materially different from those in the contemporaneous solutions. Taken together, these results indicate that Ansatz is capable not only of making partial progress on difficult open problems, but also of independently reaching complete solutions, demonstrating its strong capability in real-world mathematical research.

To enable independent verification of these claims and facilitate further study, we will publicly release the complete run artifacts and results for all ten problems, including the retrieval and tool-use records, intermediate outputs, and final solutions. This release will make it possible to inspect not only the final mathematical results but also the computational provenance of each run, including the information retrieved and used during problem solving. We believe that this will make our results transparent, publicly accessible, and reproducible.

5 Conclusion

In this paper, We present Ansatz, a mathematical research agent built around Continual Graph Memory, a graph-based and evolvable memory system for persistent mathematical reasoning. By preserving and organizing intermediate progress, failed attempts, and reusable insights, Ansatz supports sustained proof search across long reasoning trajectories. Across the ten *First Proof Second Batch* problems, Ansatz closes all ten tasks. It also produces solutions to the Jamison caterpillar conjecture and Erdős Problems 289, 348, and 488 without human intervention, showing the potential of structured research memory beyond benchmark problems. Our results suggest that representing the proof-search process in a graph-based memory, enabling cross-problem learning, and continuously evolving the memory provide an effective way to support long-horizon mathematical problem solving and move mathematical research agents toward sustained exploration.

AI USE STATEMENT

Generative AI models are central components of Ansatz. During our experiments, they were used to formulate mathematical claims, generate critical ingredients and candidate proofs, and critique and verify mathematical arguments, as described in Sections 3–4. We also used generative AI tools to support method implementation and translation, create and modify software code, summarize and analyze existing literature, and retrieve or search for relevant information. All AI-assisted outputs reported as research claims or artifacts were independently reviewed by the authors. The authors take full responsibility for the final content of this work, including all text, claims, and artifacts produced with the assistance of generative AI tools.

ETHICS STATEMENT

This paper primarily focuses on agent systems for mathematical research. There are several potential societal consequences of our work, none of which we feel must be specifically highlighted here.

REPRODUCIBILITY STATEMENT

The system developed in this work is publicly accessible online, and references are provided for all baseline methods considered in our experiments. Further technical details regarding the implementation and experimental setup are provided in the Appendix.

REFERENCES

Mohammed Abouzaid, Nikhil Srivastava, Rachel Ward, and Lauren Williams. First proof second batch, 2026. URL https://arxiv.org/abs/2606.18119.

Maria Axenovich and Felix Christian Clemen. Rainbow subgraphs in edge-colored complete graphs: answering two questions by Erdős and Tuza. *Journal of Graph Theory*, 2024. Also arXiv:2209.13867.

Abdul Basit and David Galvin. On the independent set sequence of a tree. *Electronic Journal of Combinatorics*, 28(3), 2021. arXiv:2006.12562.

Maciej Besta, Nils Blach, Aleš Kubíček, Robert Gerstenberger, Michał Podstawski, Lukas Gianinazzi, Joanna Gajda, Tomasz Lehmann, Hubert Niewiadomski, Piotr Nyczyk, and Torsten Hoefler. Graph of thoughts: Solving elaborate problems with large language models. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pp. 17682–17690, 2024. doi: 10.1609/aaai.v38i16.29720. URL https://ojs.aaai.org/index.php/AAAI/article/view/29720.

Thomas F. Bloom. Erdős Problem #1063. https://www.erdosproblems.com/1063, 2026a. Accessed 26 September 2026.

Thomas F. Bloom. Erdős Problem #289. https://www.erdosproblems.com/289, 2026b. Accessed 25 September 2026.

Thomas F. Bloom. Erdős Problem #302. https://www.erdosproblems.com/302, 2026c. Accessed 26 September 2026.

Thomas F. Bloom. Erdős Problem #348. https://www.erdosproblems.com/348, 2026d. Accessed 26 September 2026.

Thomas F. Bloom. Erdős Problem #488. https://www.erdosproblems.com/488, 2026e. Accessed 26 September 2026.

Thomas F. Bloom. Erdős Problem #561. https://www.erdosproblems.com/561, 2026f. Accessed 25 September 2026.

Thomas F. Bloom. Erdős Problem #644. https://www.erdosproblems.com/644, 2026g. Accessed 25 September 2026.

Thomas F. Bloom. Erdős Problem #811. https://www.erdosproblems.com/811, 2026h. Accessed 25 September 2026.

Thomas F. Bloom. Erdős Problem #993. https://www.erdosproblems.com/993, 2026i. Accessed 24 September 2026.

Stefan A. Burr, Paul Erdős, Ralph J. Faudree, Cecil C. Rousseau, and Richard H. Schelp. Ramsey-minimal graphs for multiple copies. *Indagationes Mathematicae*, 40:187–195, 1978.

Prateek Chhikara, Dev Khant, Saket Aryan, Taranjeet Singh, and Deshraj Yadav. Mem0: Building production-ready AI agents with scalable long-term memory, 2025. URL https://arxiv.org/abs/2504.19413.

Felix Christian Clemen and Adam Zsolt Wagner. Balanced edge-colorings avoiding rainbow cliques of size four. *Electronic Journal of Combinatorics*, 30(3):#P3.17, 2023.

Akbar Davoodi, Ramin Javadi, Azam Kamranian, and Ghaffar Raeisi. On a conjecture of Erdős on size Ramsey number of star forests. *Ars Mathematica Contemporanea*, 2025. Paper No. 9.

Prakhar Dixit and Tim Oates. ISM: Self-improving strategy memory for continual mathematical reasoning, 2026. URL https://arxiv.org/abs/2606.31191.

Erdős Problem a Day. #1063: A real upper-bound jump on n_k—corrected after a reader caught us overclaiming it. https://erdosproblemaday.com/day/1063-one-large-prime, 2026. AI-generated report, 25 July 2026.

Paul Erdős and Ronald L. Graham. *Old and New Problems and Results in Combinatorial Number Theory*, volume 28 of *Monographies de L'Enseignement Mathématique*. L'Enseignement Mathématique, Geneva, 1980.

Paul Erdős and John L. Selfridge. Problem 6447. *The American Mathematical Monthly*, pp. 710, 1983.

Paul Erdős and Zsolt Tuza. Rainbow subgraphs in edge-colorings of complete graphs. In *Quo Vadis, Graph Theory?*, volume 55 of *Annals of Discrete Mathematics*, pp. 81–88. North-Holland, 1993.

Paul Erdős, Dmitry Fon-Der-Flaass, Alexandr V. Kostochka, and Zsolt Tuza. Small transversals in uniform hypergraphs. *Siberian Advances in Mathematics*, 2:82–88, 1992.

Dmitry G. Fon-Der-Flaass, Alexandr V. Kostochka, and Douglas R. Woodall. Transversals in uniform hypergraphs with property $(7, 2)$. *Discrete Mathematics*, 207:277–284, 1999.

Ting Gong, Michael Ruofan Zeng, and Yong Yang. Albilich: Steerable proof-state orchestration for LLM-based mathematical research with CAS integration, 2026. URL https://arxiv.org/abs/2607.27705.

Robert E. Jamison. On the average number of nodes in a subtree of a tree. *Journal of Combinatorial Theory, Series B*, 35:207–223, 1983.

Robert E. Jamison. Monotonicity of the mean order of subtrees. *Journal of Combinatorial Theory, Series B*, 37:70–78, 1984.

Haocheng Ju, Guoxiong Gao, Jiedong Jiang, Bin Wu, Zeming Sun, Leheng Chen, Yutong Wang, Yuefeng Wang, Zichen Wang, Wanyi He, Peihao Wu, Liang Xiao, Ruochuan Liu, Bryan Dai, and Bin Dong. Automated conjecture resolution with formal verification, 2026. URL https://arxiv.org/abs/2604.03789.

Dmitry Khanukov. Two-sided computer-assisted progress on Erdős Problem 302. Unrefereed preprint, 2026. URL https://github.com/khanukov/erdos302/blob/907f1ebcbb0c36f3fa24749b4b722ca04e7bbb09/paper/erdos302_two_sided.tex. Version 0.2.0-preprint, 13 September 2026.

Vjekoslav Kovač. Comment on Erdős Problem #289. https://www.erdosproblems.com/forum/thread/289, 2025. 22 September 2025. Accessed 25 September 2026.

Jihao Liu, Guoxiong Gao, Zeming Sun, Bin Wu, Shurui Liu, Jiedong Jiang, Haocheng Ju, Leheng Chen, Ronnie Cheng, Xiping Zhang, and Bin Dong. Danus: Orchestrating mathematical reasoning agents with fact-graph memory, 2026. URL https://arxiv.org/abs/2607.06447.

Lucas Mol and Ortrud R. Oellermann. Maximizing the mean subtree order. *Journal of Graph Theory*, 91:326–352, 2019.

Jean-Marie Monier. Problems and solutions: Solutions of advanced problems: 6447. *The American Mathematical Monthly*, pp. 435–436, 1985.

Charles Packer, Sarah Wooders, Kevin Lin, Vivian Fang, Shishir G. Patil, Ion Stoica, and Joseph E. Gonzalez. MemGPT: Towards LLMs as operating systems, 2023. URL https://arxiv.org/abs/2310.08560.

rickyc. Comment on Erdős Problem #1063. https://www.erdosproblems.com/forum/thread/1063, 2026. 26 June 2026. Accessed 26 September 2026.

Stephen Robertson and Hugo Zaragoza. The probabilistic relevance framework: BM25 and beyond. *Foundations and Trends in Information Retrieval*, 3(4):333–389, 2009. doi: 10.1561/1500000019. URL https://www.nowpublishers.com/article/DownloadEBook/INR-019.

Noah Shinn, Federico Cassano, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. Reflexion: Language agents with verbal reinforcement learning. In *Advances in Neural Information Processing Systems*, volume 36, 2023. URL https://papers.neurips.cc/paper_files/paper/2023/hash/1b44b878bb782e6954cd888628510e90-Abstract-Conference.html.

Guanzhi Wang, Yuqi Xie, Yunfan Jiang, Ajay Mandlekar, Chaowei Xiao, Yuke Zhu, Linxi Fan, and Anima Anandkumar. Voyager: An open-ended embodied agent with large language models, 2023a. URL https://arxiv.org/abs/2305.16291.

Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc Le, Ed Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. Self-consistency improves chain of thought reasoning in language models. In *International Conference on Learning Representations*, 2023b. URL https://arxiv.org/abs/2203.11171.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed Chi, Quoc Le, and Denny Zhou. Chain-of-thought prompting elicits reasoning in large language models. In *Advances in Neural Information Processing Systems*, volume 35, 2022. URL https://arxiv.org/abs/2201.11903.

Wujiang Xu, Zujie Liang, Kai Mei, Hang Gao, Juntao Tan, and Yongfeng Zhang. A-Mem: Agentic memory for LLM agents. In *Advances in Neural Information Processing Systems*, volume 38, 2025. URL https://papers.neurips.cc/paper_files/paper/2025/hash/19909c36f51abc4856b4560aff3d36d6-Abstract-Conference.html.

Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Thomas L. Griffiths, Yuan Cao, and Karthik Narasimhan. Tree of thoughts: Deliberate problem solving with large language models. In *Advances in Neural Information Processing Systems*, volume 36, 2023a. URL https://arxiv.org/abs/2305.10601.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. ReAct: Synergizing reasoning and acting in language models. In *International Conference on Learning Representations*, 2023b. URL https://arxiv.org/abs/2210.03629.

Guibin Zhang, Muxin Fu, Kun Wang, Frank Wan, Miao Yu, and Shuicheng Yan. G-Memory: Tracing hierarchical memory for multi-agent systems. In *Advances in Neural Information Processing Systems*, volume 38, 2025. URL https://papers.neurips.cc/paper_files/paper/2025/hash/136a45cd9b841bf785625709a19c6508-Abstract-Conference.html.

Hanrong Zhang, Shicheng Fan, Henry Peng Zou, Yankai Chen, Zhenting Wang, Jiayu Zhou, Chengze Li, Wei-Chieh Huang, Yifei Yao, Kening Zheng, Xue Liu, Xiaoxiao Li, and Philip S. Yu. CoEvoSkills: Self-evolving agent skills via co-evolutionary verification, 2026. URL https://arxiv.org/abs/2604.01687.

A Formal specification of Continual Graph Memory

This appendix collects the definitions, update rules, and algorithms behind the description in Section 3. Each equation describes what the implementation stores or computes, and the surrounding text records the boundary conditions and defaults that the main text omits. The subsections follow the order of Section 3.

A.1 Memory state and memory-conditioned execution

For project p_j at event t, the persistent state is

$$\mathcal{M}_{j,t} = \big(\{L_{i,t}\}_{i=1}^{m},\ C_t,\ F_t,\ E_t^F,\ E_t^X,\ A_t\big). \tag{A.1}$$

Here L_i is worker i's private process log, C contains shared findings and their evidence, F indexes accepted proof records, and E^F records their declared dependencies. The exploration overlay (E^X, A) connects plans, findings, facts, and goals through typed relations and curator annotations. Local and shared logs are append-only; a deployment-level lesson sequence Λ_j persists beyond any one project. Shared discussion can contain tentative conclusions, counterexamples, plans, or obstacles; membership in C does not imply membership in F.

An active role r chooses actions using its contract, private interaction history $h_{j,t}^r$, and memory responses $x_{j,t}^r$ that it has obtained through its permitted tools:

$$\begin{aligned} a_{j,t}^r &\sim \pi_\theta^r\big(\cdot \mid g_j, h_{j,t}^r, x_{j,t}^r, \mathrm{Contract}^r\big)\,, \\ (\mathcal{M}_{j,t+1}, o_{j,t+1}) &= \mathrm{Execute}(\mathcal{M}_{j,t}, a_{j,t}^r). \end{aligned} \tag{A.2}$$

This notation describes tool-mediated execution: π_θ^r is the role's LLM policy, g_j the task specification, and o the resulting observation. The model parameters θ remain fixed within a deployment; experimental deployments may use different base models. Algorithm 1 summarizes the project sequence, and Algorithm 2 specifies the memory operations it calls. Workers and the curator run asynchronously, so the event loop is a procedural summary rather than a synchronized round schedule.

Algorithm 1 Continual search with persistent graph memory

Require: Projects $(p_j, g_j)_{j=1}^J$, budgets, memory mode σ, lesson sequence Λ_1
Require: Operator-controlled project visibility and termination
Ensure: Project memories $\{\mathcal{M}_j\}$ and updated lessons Λ_{J+1}

for $j = 1, \ldots, J$ **do**
  Establish eligible project scope $\mathcal{U}_\sigma(p_j)$ (A.14)
  Initialize project stores and launch workers, main agent, and curator
  Initialize curator with its contract and eligible recent lessons (A.17)
  while the main agent continues the run **do**
    Answer worker memory requests with ReadMemory(q, u, p_j, σ)
    Workers explore assigned lanes and append findings and typed relations
    Process submitted proofs with Admit$(f, \mathrm{source_id})$
    On curator wake, inspect changed evidence and update affected annotations
    Refresh statuses and frontier under the curator contract (A.8)–(A.10)
    Main agent adjusts lanes, consultation, and stopping decisions
  end while
  At orderly close, optionally append validated retrospective lessons (A.16)
  Persist Λ_{j+1}; establish the next project's visibility boundary
end for

A.2 Fact identifiers, admission, and revocation

A candidate $f = (p, S, \Pi, P, G, B)$ contains a statement S, proof Π, predecessor identifiers P, glossary entries G, and bibliography metadata B. The fact store assigns

$$\mathrm{id}(f) = \mathrm{prefix}_{64}(\mathrm{SHA256}(\mathrm{canon}(p, P, G, S, \Pi)))\,. \tag{A.3}$$

The identifier uses the first 64 hash bits. Canonicalization sorts predecessors and glossary entries and normalizes statement/proof whitespace, giving repeated canonical content the same identifier. Bibliography metadata is excluded so that citation corrections preserve dependency links; citation keys inside the proof remain hashed.

Workers submit through `fact_submit`. Let $h_t(f)$ indicate that deterministic prechecks and the revoked-predecessor check pass, $v_t(f)$ be the fresh verifier's verdict, and $w_t(f)$ indicate a completed fact write. Unexecuted or failed stages take a nonaccepting value. Acknowledged admission is

$$a_t(f) = \mathbf{1}[h_t(f) = 1]\,\mathbf{1}[v_t(f) = \texttt{correct}]\,\mathbf{1}[w_t(f) = 1]. \tag{A.4}$$

For an admitted candidate with $u = \mathrm{id}(f)$, the update is

$$F_{t+1} = F_t \cup \{u\}, \qquad E^F_{t+1} = E^F_t \cup \{(r, u) : r \in P\}. \tag{A.5}$$

An edge (r, u) records a declared proof dependency. Deterministic checks reject vacuous content and selected unsupported proof patterns; glossary coverage is advisory. Once a verifier verdict exists, the gateway records the verdict, report, and any write failure in shared memory, and successful writes can link back to an originating finding. An accepting verdict without a returned fact identifier is not acknowledged admission. Equation (A.5) describes completed writes, not recovery from arbitrary I/O failures.

Proposition 1 (Local admission property)**.** *Starting from an empty fact store, suppose all additions use the default gateway and persistence succeeds. Every admitted fact has an accepting verdict from a separate verification invocation. Changing curator scores alone cannot admit a fact.*

The property follows from the gateway's write condition and role-specific tool exposure. It provides an auditable admission boundary, not a proof of verifier soundness. The implementation rejects revoked predecessors before verification and again at insertion, but does not require all predecessor identifiers to resolve or enforce acyclicity. A dependency-closed DAG additionally requires a valid local citation protocol.

If an accepted result is later challenged, revoking u removes its recorded descendants:

$$R_t(u) = \{u\} \cup \mathrm{Desc}_{E^F_t}(u), \qquad F_{t+1} = F_t \setminus R_t(u). \tag{A.6}$$

A revocation log preserves the decision. This repairs the recorded dependency structure; omitted dependencies remain outside the cascade.

A.3 Annotations, curator contract, staleness, and frontier ranking

An annotation $a = (v, z, s, e, c, \tau)$ records a node v, status z, optional score $s \in [0, 1]$, evidence identifiers e, rationale c, and timestamp τ. Its current view is

$$\begin{aligned} \widehat{A}_t(v) &= \mathrm{last}_{\mathrm{append}}\{a \in A_t : a.\mathrm{node} = v\}, \\ (z_t(v), s_t(v)) &= (\widehat{A}_t(v).z, \widehat{A}_t(v).s). \end{aligned} \tag{A.7}$$

An absent status defaults to `open`, and an absent score remains unset ($\perp$). The final appended event wins; timestamps do not select the current annotation, and earlier scores are not averaged.

The curator interprets exploration relations using an AND–OR contract. Let $\mathcal{O}(r)$ be the obligations of an AND route and $\mathrm{cl}_t(v)$ mean that v is marked closed. Representative rules are

$$\begin{aligned} \bigwedge_{v \in \mathcal{O}(r)} \mathrm{cl}_t(v) &\Longrightarrow \text{route } r \text{ closes its parent}, \\ \mathrm{cl}_t(Q) \;\wedge\; \mathrm{bridge}_t(Q \Rightarrow P) &\Longrightarrow \mathrm{cl}_t(P). \end{aligned} \tag{A.8}$$

For localization, $\mathcal{O}(r)$ includes reassembly: solving local pieces alone does not close the theorem. A reduction requires an accepted bridge; equivalence requires both directions, and generalization requires a specialization argument. Failed OR branches leave other routes available. These are curator instructions, not a symbolic inference engine. The store validates relation types and score ranges; mathematical witnessing and propagation remain agent responsibilities.

The default frontier $\mathcal{V}_t$ contains open or active plans, directions, and annotated symbolic goals. New incident edges identify judgments needing reconsideration:

$$\mathrm{stale}_t(v) = \mathbf{1}\left[\max_{e \ni v} \tau(e) > \tau\big(\widehat{A}_t(v)\big)\right]. \tag{A.9}$$

Missing timestamps and an empty maximum take value $-\infty$. For incident-edge count $d_t^X(v)$, define $\bar{s}_t(v) = 0$ if $s_t(v) = \perp$ and $\bar{s}_t(v) = s_t(v)$ otherwise. The implemented ranking is

$$\begin{aligned} \kappa_t(v) &= \big(\mathbf{1}[d_t^X(v) = 0], \mathbf{1}[s_t(v) = \perp], -\bar{s}_t(v), c_t(v)\big), \\ \mathrm{Frontier}_K(t) &= \mathrm{first}_K\, \mathrm{sort}_{\mathrm{lex}}(\mathcal{V}_t; \kappa_t). \end{aligned} \tag{A.10}$$

Linked nodes precede unlinked nodes, scored nodes precede unscored nodes, and higher scores precede lower scores. Ties favor older entries; symbolic goals use their latest annotation time for $c_t(v)$. The tool exposes a ranked worklist, while the curator supplies its judgments. The contract asks for an exploration bonus and permits strategic guidance to override ranking; no fixed UCB equation or learned scoring model is implemented. Staleness detects newer edges rather than semantic changes to evidence.

A.4 Lexical scoring and neighborhood retrieval

For tokenized corpus $\mathcal{D}$ of size N, query q, and document d, the implemented BM25 (Robertson & Zaragoza, 2009) score is

$$\begin{aligned} \mathrm{BM25}(q, d; \mathcal{D}) &= \sum_{x \in \mathrm{uniq}(q)} n(x, q)\, \mathrm{idf}(x) \frac{(k_1 + 1) n(x, d)}{n(x, d) + k_1(1 - b + b|d|/\overline{L})}, \\ \mathrm{idf}(x) &= \log\left(1 + \frac{N - \mathrm{df}(x) + \frac{1}{2}}{\mathrm{df}(x) + \frac{1}{2}}\right), \qquad \overline{L} = N^{-1} \sum_{d \in \mathcal{D}} |d|. \end{aligned} \tag{A.11}$$

Here $n(x, \cdot)$ counts occurrences, $\mathrm{df}(x)$ counts documents containing x, $k_1 = 1.5$, and $b = 0.75$. Query multiplicity therefore matters. Tokenization lowercases ASCII letters, digits, and underscores. Empty queries or corpora score zero; an all-empty corpus uses normalization term k_1. Fact search indexes stored Markdown and returns positive-scoring identifiers and statements. Agents read proofs separately, and indexes are rebuilt from stored content on demand.

For a selected fact identifier u, `fact_neighbors` retrieves declared predecessors and nearby exploration facts. With $u \sim_X v$ denoting an exploration edge in either direction, the eligible overlay set is

$$H_t(u) = \big\{g \in F_t \setminus \{u\} : u \sim_X g \ \vee\ \exists v \notin F_t,\ u \sim_X v \sim_X g\big\}. \tag{A.12}$$

Traversal is undirected; returned paths preserve edge labels and directions for interpretation, without asserting entailment.

Let $P_B(u) = \mathrm{first}_B(P(u))$ preserve the stored predecessor order, and let $H_r^{\mathrm{scan}}(u)$ select the first r distinct overlay facts in edge-log traversal order. The default result sequence is

$$\begin{aligned} B &= \max(1, \mathrm{limit}), \qquad r = B - |P_B(u)|, \\ \mathcal{N}_B(u) &= P_B(u) \parallel \big[g \in H_r^{\mathrm{scan}}(u) : g \notin P_B(u)\big], \qquad \mathrm{limit} = 25. \end{aligned} \tag{A.13}$$

Here $\parallel$ concatenates sequences, and no overlay results are added when the budget is exhausted. Predecessors receive priority; the overlay cap is applied before predecessor duplicates are removed, so the result can underfill its budget. Optional immediate dependents consume budget between predecessors and overlay facts, and their count is always reported. These operations return statements and connecting paths; they leave proof inspection and local admission to the agent.

A.5 Scope, scoped recall, and lesson continuation

Let $\mathcal{P}$ denote available projects, $\beta(p)$ their optional batch labels, and σ the memory mode. Fact statements and negative findings are retrieved from

$$\mathcal{U}_\sigma(p) = \begin{cases} \varnothing, & \sigma = \texttt{off}, \\ \{p' \in \mathcal{P} \setminus \{p\} : \beta(p') = \beta(p) \neq \perp\}, & \sigma = \texttt{batch}, \\ \mathcal{P} \setminus \{p\}, & \sigma = \texttt{all}. \end{cases} \tag{A.14}$$

Algorithm 2 Memory access and proof admission primitives

procedure READMEMORY(q, u, p, σ)
  $Q \leftarrow$ requested project-local finding and fact search results
  if the request includes a known fact identifier u **then**
    Add $\mathcal{N}_B(u)$ with statements and paths to Q (A.13)
  end if
  if cross-project recall is requested and $\sigma \neq$ `off` **then**
    Build eligible per-channel corpora for project p (A.14)
    Add $\{\mathcal{R}_c(q)\}_{c\in\mathcal{C}}$ with source provenance to Q (A.15)
  end if
  return Q; read full records separately when needed
end procedure
procedure ADMIT(f, source_id)
  Check revoked predecessors and deterministic preconditions
  if checks fail **then**
    return error and available repair information
  end if
  Obtain a verdict and report from a fresh verifier invocation
  if the verifier call fails or returns a non-dictionary **then**
    return verification error
  end if
  $u \leftarrow \perp$; initialize write-error field
  if verdict is `correct` **then**
    Attempt the fact write, rechecking revoked predecessors (A.5)
    On success set $u \leftarrow \mathrm{id}(f)$; otherwise retain the write error
  end if
  Append verdict, report, identifier u, and write error to shared memory
  if $u \neq \perp$ and a source finding was supplied **then**
    Attempt an advisory promotion link from the finding to u
  end if
  return verdict, identifier, and repair or error information
end procedure

The gateway defaults to `off`, including for invalid mode values. Batch membership establishes visibility, not chronology. A sequential evaluation additionally requires $\mathcal{U}_\sigma(p_j) \subseteq \{p_1, \ldots, p_{j-1}\}$, and earlier projects must stop changing at their designated boundary. The operator must enforce these conditions through project visibility and termination; recall itself has no completion or timestamp filter.

Recall searches fact statements, negative findings (dead ends, obstacles, and counterexamples), and methodological lessons separately. For requested channels $\mathcal{C}$ and eligible corpus $\mathcal{D}_c$, it returns

$$\begin{aligned} \mathcal{R}_c(q) &= \mathrm{Top}_k^+\big\{(d, \mathrm{BM25}(q, d; \mathcal{D}_c)) : d \in \mathcal{D}_c\big\}, \\ k &= \max\left(1, \left\lfloor \frac{\text{limit}}{|\mathcal{C}|} \right\rfloor\right). \end{aligned} \tag{A.15}$$

Top^+ retains only positive scores. The default budget gives five results to each of three channels; scores are corpus-specific and are not jointly normalized. In `batch` mode, queryable lessons may come from $\mathcal{U}_\sigma(p) \cup \{p\}$; in `all` mode, the deployment lesson log is unfiltered. Thus a current-project lesson can be recalled even though current-project facts and negative findings are excluded from this tool. Returned facts contain statements rather than proofs, retain their source provenance, and require local reproof and admission under the worker contract. Strategic guidance and progress elaborations are excluded from recall.

At orderly project close, the curator may record $\ell = (\mathrm{pattern}, \mathrm{evidence}, \mathrm{adjustment})$: a recurring scoring habit, supporting outcome evidence, and a proposed adjustment. Each text field must satisfy

$$\mathrm{Valid}(\ell) = \bigwedge_{x\in\ell} [\,|\,\mathrm{strip}(x)| > 0 \ \wedge\ |x| \leq 2000\,]\,. \tag{A.16}$$

The store strips validated fields and appends author, source project, and timestamp. The contract limits lessons to methodology; no semantic filter enforces that restriction. For optional retrospective $\Delta\Lambda_j$, the next curator starts with

$$\begin{aligned} \Lambda_{j+1} &= \Lambda_j \| \Delta\Lambda_j, \qquad \theta_{j+1} = \theta_j, \\ \mathcal{I}_{j+1}^{\text{curator}} &= \text{Contract} \| \operatorname{last}_{20}\big(\text{Eligible}_{\sigma}^{\text{launch}}(\Lambda_{j+1}, p_{j+1})\big). \end{aligned} \tag{A.17}$$

Launch eligibility selects no lessons in `off`, sources from the same nonempty batch in `batch`, and all lessons in `all`; recency follows append order. Workers and the main agent query lessons, while the curator receives this initial context. Retrospectives are optional and can be lost at abrupt termination, so persistent memory does not by itself imply improvement across tasks.

B Statements of Partially Solved Open Problems

811. An edge-colouring of K_n with $m = e(G)$ colours is balanced when $n \equiv 1 \pmod m$ and every vertex meets every colour equally often. Erdős, Pyber and Tuza asked which graphs G appear as rainbow subgraphs in every large balanced colouring (Erdős & Tuza, 1993; Bloom, 2026h). Axenovich and Clemen showed that most cliques fail, and conjectured that K_q fails for every $q \geq 4$ (Axenovich & Clemen, 2024). Clemen and Wagner settled $q = 4$ (Clemen & Wagner, 2023). Ansatz gives a finite balanced base colouring with no rainbow K_q for every $q \geq 4$, then lifts it to arbitrarily large admissible orders by the amplification of Axenovich & Clemen (2024). For $q \geq 12$ the base is a pairing of cyclic distances on $\mathbb{Z}_N$ with $N = 4\binom{q}{2} + 1$. The eight bases with $q \leq 11$ were checked by exhaustive search. This settles every clique. The question for a general graph, including C_6, stays open.

644. Let $f(k, 7)$ be the largest transversal number of a k-uniform family in which every seven members have a two-point transversal. The conjectured order is $\frac{3}{4}k$ (Erdős et al., 1992; Bloom, 2026g). Fon-Der-Flaass, Kostochka and Woodall proved $f(k, 7) \leq \lceil 7k/8 \rceil$ (Fon-Der-Flaass et al., 1999). For $k \geq 150$, Ansatz gives $f(k, 7) \leq \lceil (6k + 3)/7 \rceil + 1$, hence $\limsup f(k, 7)/k \leq 6/7$.

561. For star forests $F_1 = \bigcup_i K_{1,n_i}$ and $F_2 = \bigcup_j K_{1,m_j}$, write L for the sum of the diagonal maxima $l_k = \max\{n_i + m_j - 1 : i + j = k\}$. Burr, Erdős, Faudree, Rousseau and Schelp conjectured $\hat{R}(F_1, F_2) = L$ (Burr et al., 1978; Bloom, 2026f). The formula was known for uniform forests, for a single star, for two equal stars, and when every size is odd (Davoodi et al., 2025). Ansatz proves it when one side is two stars of arbitrary sizes and the other side is an arbitrary star forest.

993. Erdős asked whether the independence sequence $i_k(T)$ of every tree, or forest, is unimodal (Bloom, 2026i). Basit and Galvin showed the sequence is nondecreasing through roughly $n/8$ for trees (Basit & Galvin, 2021). Ansatz produced a proof that every forest increases strictly while through the first $\frac{131}{520}n$ terms, lifting the increasing prefix from Basit and Galvin's $n/8$ to slightly past $n/4$. For forests of maximum degree at most four it proved strict increase through $0.276\,n$, which is sharp since the path stops exactly there. The conjecture itself remains open.

1063. Let n_k be the least $n \geq 2k$ for which exactly $k-1$ of $n, n-1, \ldots, n-k+1$ divide $\binom{n}{k}$ (Erdős & Selfridge, 1983; Bloom, 2026a). Monier proved $n_k \leq k!$ for $k \geq 3$, and Cambie improved this to $n_k \leq \exp((1 + o(1))k)$ (Monier, 1985; Bloom, 2026a). The catalogue discussion also records lower bounds depending on the prime factorization of k (rickyc, 2026). A later AI-generated report claims $n_k \leq \exp(Ck \log\log k / \log k)$ for an unspecified constant C (Erdős Problem a Day, 2026). For sufficiently large k, Ansatz proves

$$\begin{aligned} n_k &> \exp\big(10^{-30}(\log k)^2\big), \\ n_k &\leq \exp\left(\frac{k}{\log k}\left[\log\log k + \log\log\log k + \log 2 - \eta + o(1)\right]\right), \end{aligned}$$

where $\eta > 0$ is an absolute constant from the Fourier construction. The lower argument combines primes in short intervals with rational approximation; the upper argument constructs admissible integers using nonnegative Fourier coefficients. These bounds give a super-polynomial lower bound and refine the previously claimed upper bound. A matching growth estimate remains open.

302. Let $f(N)$ be the largest size of a subset of $\{1, \dots, N\}$ containing no distinct a, b, c with $1/a = 1/b + 1/c$ (Erdős & Graham, 1980; Bloom, 2026c). The catalogue records Cambie's asymptotic lower coefficient $5/8$ and van Doorn's upper coefficient $9/10$ (Bloom, 2026c). Khanukov's preprint gives a non-explicit positive gain over $5/8$ and an upper coefficient $140803024/163562355 < 0.860853$ (Khanukov, 2026). Ansatz proves $f(N) = cN + o(N)$, characterizes c as an infimum of approximations using finite sets of primes, and gives an explicit convergence rate. It also proves $c \geq 5/8 + \exp(-100000000030)$, making the lower gain explicit. The exact value of c remains open.