CLOSING THE LOOP IN FORMAL THEOREM PROVING: COORDINATED AND REVISABLE PROOF CONSTRUCTION

Anonymous authors
Paper under double-blind review

ABSTRACT

Agentic theorem provers increasingly combine mathematical reasoning, proof decomposition, and iterative interaction with Lean. When proof construction stalls, the underlying difficulty may arise in formal proving, the informal proof strategy, or the surrounding proof structure, and Lean's diagnostics alone do not identify which source is responsible. Effective recovery requires identifying the source of difficulty and controlling the scope of revision. We present OpenMathHarness (OMH), an agentic theorem-proving framework that coordinates decomposition, informal reasoning, formal proving, and revision within a unified proof construction process. OMH introduces a multi-level review mechanism that uses formal evidence to direct subsequent recovery effort to formal refinement, revision of the informal proof strategy under a fixed statement, or structural revision. Using DeepSeek-V4-Flash as its backbone, OMH solves 53/100 problems on FATE-X and 11/12 on Putnam 2025, at mean inference costs of $1.23 and $0.27 per solved problem respectively. Controlled experiments show that decomposition and informal guidance improve performance on effort-intensive problems, and strategy flexibility nearly halves token usage on commonly solved nodes. Analysis of review and routing decisions further shows how recovery unfolds across formal, strategy, and structural levels.

1 INTRODUCTION

Large language models (LLMs) have made rapid progress in mathematical reasoning, advancing from gold-medal performance at the International Mathematical Olympiad (Huang & Yang, 2025; Luong et al., 2025) to strong results on research-level benchmarks such as FrontierMath (Glazer et al., 2024), and recently to contributions to open problems such as Erdős problems (Feng et al., 2026; Alexeev et al., 2026) and the unit-distance problem (OpenAI, 2026b). While models can generate candidate proofs at scale, verification has become the primary bottleneck, since manual verification by human experts is slow and expensive. Formal theorem proving in a machine-checkable language such as Lean (de Moura & Ullrich, 2021) provides a solution, as kernel-accepted proofs are mechanically verified.

Automated theorem proving in such languages initially relied on symbolic systems and later on specialized language models trained on formal corpora, bringing benchmarks such as MiniF2F (Zheng et al., 2022) close to saturation (Ren et al., 2025; Lin et al., 2026). More recent work increasingly adopts agentic systems that combine proof decomposition, informal mathematical reasoning, and iterative interaction with proof assistants (Liu et al., 2026; Chen et al., 2025; Requena et al., 2026). This development turns formal theorem proving into a multi-stage process in which proof structure, mathematical strategy, and formal verification jointly shape how a proof evolves.

This organization introduces ambiguity in diagnosing stalled proof construction. A failed formal attempt may arise from an error in formal proof construction, an unsuitable proof strategy, or a structural defect such as a false or poorly scoped intermediate lemma. Lean provides local compiler diagnostics, but these signals alone do not identify which component is responsible. Existing systems provide recovery mechanisms at different stages of proof construction, but higher-level revision is often tied to stage-specific workflows or triggered after local proving reaches a predefined

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.

boundary (Varambally et al., 2026; Kung et al., 2026). As a result, recovery can be misallocated in either direction. Strategy- or structure-level difficulties may be repeatedly handled through local formal repair, while locally resolvable failures may trigger unnecessary decomposition. Recovery therefore depends on locating the source of difficulty and matching the scope of revision to it.

Appropriate routing further requires controlling the scope of revision. A mathematically valid informal strategy may nevertheless require substantial formal machinery or rely on results unavailable in Mathlib (The mathlib Community, 2020). Some systems explicitly encourage the formal prover to remain closely aligned with a preceding informal solution (Qiu et al., 2026; Achim et al., 2025), which may commit substantial effort to a route that is mathematically sound but poorly suited to the formal environment. Other systems use prover feedback to revise the surrounding proof structure (Chung et al., 2026), but structural revision may be unnecessary when the current statement remains appropriate and an alternative proof strategy is available. Separating the proof objective from the strategy used to establish it allows the system to revise how a local objective is proved without prematurely altering the surrounding proof structure.

We present OpenMathHarness (OMH), an agentic theorem-proving framework that coordinates decomposition, informal reasoning, formal proving, and revision within a persistent proof construction process (Figure 1). OMH represents proof construction as a proof graph rooted at the target theorem, with intermediate lemmas introduced through decomposition. Each intermediate lemma is associated with informal guidance that proposes a default proof strategy. The formal prover may adopt an alternative route when it can establish the same statement more effectively in Lean. When progress stalls, local review determines whether to continue the current formal trajectory or revise the guidance and restart proving while keeping the lemma statement fixed. If local recovery is insufficient, node-level review uses the surrounding proof context to decide whether to retry the current objective, further decompose it, or revise the parent decomposition. In this way, evidence produced during formal proving can reshape both the strategy for an individual lemma and the surrounding proof structure, and direct subsequent proof effort to the appropriate level indicated by the review. Our contributions are as follows.

- We present OpenMathHarness (OMH), an agentic theorem-proving system that coordinates decomposition, informal reasoning, formal proving, and revision within a unified and revisable proof construction process.
- We introduce a multi-level review mechanism that uses formal evidence to guide subsequent recovery, allowing recovery to range from local proof refinement and strategy revision under a fixed statement to changes in the surrounding proof structure while retaining verified progress.
- OMH achieves 53/100 on FATE-X, together with 92/100 on FATE-H and 11/12 on Putnam 2025, at mean inference costs of \$1.23, \$0.37, and \$0.27 among solved problems, respectively. Controlled ablations evaluate the roles of decomposition, informal guidance, and strategy flexibility. Analysis of review and routing decisions characterizes recovery patterns across formal, strategy, and structural levels.

2 Related Work

Formal theorem proving models and agentic systems. Specialized language models have advanced formal theorem proving through techniques such as domain-specific training, synthetic proof data, reinforcement learning, and self-correction (Azerbayev et al., 2024; Wang et al., 2024b; Xin et al., 2025; Ren et al., 2025; Lin et al., 2025b; 2026; Wang et al., 2025). More recent systems increasingly integrate formal proving with iterative interaction and broader proof orchestration. COPRA performs tactic-level proof search through stateful backtracking and repeated interaction with the proof assistant (Thakur et al., 2024), while AxProverBase shows that a lightweight agent built around iterative whole-proof refinement, library search, and context management can already provide a strong baseline (Requena et al., 2026). APOLLO uses Lean feedback to isolate and repair failing proof regions (Ospanov et al., 2025). Recent specialized models such as Leanstral 1.5 and LongCat-Flash-Prover further incorporate agentic interaction directly into model training, using compiler or tool feedback across multi-turn proof construction (Mistral AI, 2026; Wang et al., 2026). Beyond local iterative proving, Seed-Prover 1.5 organizes natural-language proving, sketch generation, and agentic Lean proving into a hierarchical test-time workflow (Chen et al., 2025). Prover

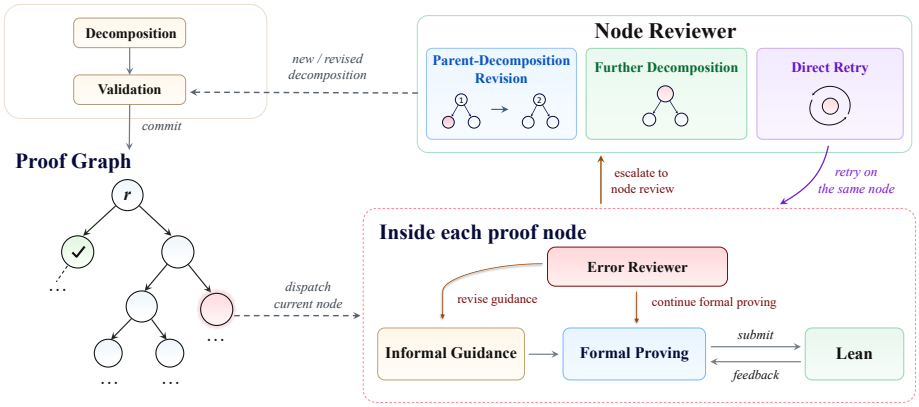

Figure 1: Overview of the OpenMathHarness (OMH) framework. OMH maintains a persistent proof graph whose current node is dispatched to a local proving loop combining informal guidance, formal proving, Lean feedback, and local review. When escalating to node review, the node reviewer selects among a direct retry, further decomposition, and parent-decomposition revision. New or revised decompositions are validated before being committed to the proof graph, closing the feedback loop between proof construction and structural revision.

Agent combines informal reasoning and formal proving with auxiliary-lemma generation (Baba et al., 2025). Numina-Lean-Agent adopts a general coding agent that autonomously coordinates Lean interaction, theorem retrieval, informal reasoning, and auxiliary tools (Liu et al., 2026). This shift makes the organization of proof construction itself an increasingly important part of system design, beyond the capability of the local formal prover.

Structured proof construction and revision. Informal reasoning has been incorporated into formal theorem proving in several ways. Draft, Sketch, and Prove translates informal proofs into formal proof sketches that guide automated proving toward simpler subproblems (Jiang et al., 2023), while Lean-STaR trains models to generate informal reasoning before individual formal proof steps (Lin et al., 2025a). A separate line of work develops reusable proof knowledge across problems. LEGO-Prover constructs proofs with a growing library of verified lemmas that can be created and reused during proving (Wang et al., 2024a), and DreamProver further abstracts and consolidates discovered lemmas into a transferable library through a wake-sleep process (Zhang et al., 2026). More recent systems make the structure of an individual proof process explicit. Hilbert recursively decomposes difficult goals into subgoals using informal reasoning and formal verification (Varambally et al., 2026). Delta Prover combines reflective decomposition with iterative proof repair (Zhou et al., 2025). Mechanic derives decomposition directly from partially successful formal proofs (Qiu et al., 2026). LEAP maintains hierarchical decompositions in an AND-OR proof graph, combining informal blueprints with iterative formal refinement (Kung et al., 2026). Goedel-Architect takes a more global approach by constructing a dependency blueprint of intermediate definitions and lemmas, and revising this blueprint when downstream lemma proving fails (Chung et al., 2026). These works move formal theorem proving toward increasingly structured and iterative proof construction. OMH focuses specifically on how downstream formal evidence should affect earlier proof decisions. It separates the statement of a lemma, the strategy used to prove it, and the surrounding proof structure, and directs proof effort to the appropriate level implied by the evidence.

3 METHODS

OMH organizes proof construction over a persistent proof graph and employs a multi-level review mechanism to guide recovery when proving stalls. The error reviewer operates locally under a fixed lemma statement, deciding whether to continue formal refinement or adjust the informal guidance and its underlying proof strategy. Keeping the statement fixed separates how the lemma is proved

from its role in the surrounding proof structure. When local recovery is insufficient, control passes to a node reviewer, which determines whether to make another direct attempt, further decompose the current lemma, or revise the parent decomposition. The complete proof construction process is summarized in Algorithm 1.

3.1 Proof Graph Structure

Given a target theorem, OMH maintains a proof graph whose root represents the theorem and whose other nodes represent auxiliary lemmas. Each node carries a formal statement and informal guidance. Both the root and auxiliary lemmas can be proved directly or decomposed further. A decomposition specifies child lemmas and an assembly proof that derives the parent goal from the required children. Lemmas may be shared across decompositions, allowing verified results to be reused. The graph therefore provides a persistent representation of the proof, where subsequent procedures may extend or revise selected portions of the structure while retaining verified nodes that remain relevant.

Decomposition occurs both during initial planning and during subsequent proof construction. The planner may select direct proving for a sufficiently simple target, avoiding the overhead of decomposition. Otherwise, it proposes an informal decomposition plan. A plan reviewer evaluates whether the proposed lemmas are mathematically valid, meaningfully reduce the difficulty of the proof, and collectively suffice to establish the target theorem. Subsequent decompositions are determined during proof construction. Specifically, the node reviewer may request further decomposition of the current goal or revision of its parent decomposition (Section 3.3). A decomposer then proposes a new or revised decomposition, specifying the child lemmas and an informal assembly argument for deriving the parent goal. The resulting proposal is reviewed under the same validity, simplification, and sufficiency criteria as the initial plan.

Once accepted by the corresponding reviewer, both initial and subsequent decomposition proposals enter a common formalization and validation process. A formalizer produces Lean statements for the child lemmas and an assembly proof for the parent. At each decomposition step, only the immediate child lemmas are formalized. Deeper ones remain informal until the corresponding child is selected for further expansion. OMH checks the decomposition in Lean using `sorry` placeholders for the child lemmas, which verifies that their statements elaborate and that the assembly proof derives the parent goal from the child lemmas. A faithfulness reviewer then assesses whether the formal lemma statements preserve their intended meaning in the informal decomposition. The decomposition is added to the proof graph only after passing both the Lean checks and faithfulness review. This validation separates the structural correctness, whether the parent node follows from the proposed child lemmas, from the subsequent task of constructing verified proofs of those lemmas.

Node goals are dispatched in parallel with their informal guidance and available verified facts. Each unresolved node may be proved directly or further decomposed through the same process. A node is considered proved only when a complete proof of its fixed statement passes Lean verification. For a decomposed node, its required child lemmas are first established, after which the parent assembly is verified using their proofs. This process proceeds recursively until the root theorem is established. The final assembled proof of the root theorem is then verified independently before being accepted as complete. Further details of the verification procedure are provided in Appendix B.1.

3.2 Local Proving and Review

For each dispatched node, we call one bounded local proving loop a round. A local attempt consists of an initial round and up to three returns. In each round, the informal prover first develops the supplied guidance into a more detailed informal proof sketch. The sketch is reviewed before being passed to the formal prover, providing a strategy for proving the current lemma. The formal prover is equipped with LeanSearch (Gao et al., 2026) for retrieving relevant library results and a Lean compilation tool for checking and iteratively refining proof candidates. During this process, compiler feedback is returned to the formal prover for subsequent proof construction. A candidate that is submitted by the formal prover and successfully compiles is subjected to separate Lean verification before the node is accepted as proved.

The informal guidance provides a default proof strategy, but a mathematically valid strategy may be difficult to realize using the available Lean library. The formal prover is therefore allowed to

adopt an alternative strategy while preserving the statement and its hypotheses, without revising the surrounding decomposition. Any departure from the default proof strategy must be explicitly declared, with a brief explanation of the alternative strategy and the reason for the change. We refer to this declared departure, together with guidance revision by the error reviewer described below, as strategy flexibility.

When a formal proving loop stops at a checkpoint without a verified proof, the error reviewer examines the statement, informal guidance, formal candidate, compiler diagnostics, and prior recovery attempts to determine where further effort is warranted. Local review distinguishes between continuing the current formal trajectory and restarting local proving under revised guidance. If the current strategy remains viable and the formal proof is close to completion, the error reviewer authorizes a continuation when further refinement is likely to succeed, preserving the progress already made. Such returns are bounded to prevent indefinite iteration.

A different form of recovery is required when the difficulty is directed to the current guidance or proof strategy. Within an attempt loop, accumulated errors and repeated unsuccessful repairs may cause the prover to become trapped in a proof strategy that is difficult to formalize, even though it has already been allowed to change proof strategy. In such cases, the error reviewer can request a revision of the informal guidance and initiate a fresh attempt, allowing the prover to explore a new strategy without inheriting the failed trajectory of the previous attempt. The same recovery path applies when the informal guidance contains a mathematical error. The guidance is corrected with the lemma statement fixed, after which formal proving restarts from the revised informal guidance.

The third choice of local review is escalation, which may occur when the available evidence suggests that the difficulty lies beyond the scope of recovery within the current node, for example when the statement itself appears false or poorly scoped, or when the decomposition is suspected to be defective. It can also be triggered once the recovery budget is exhausted. In such cases, the error reviewer forwards the node to the node reviewer for higher-level reconsideration. This escalation path allows evidence gathered during local proving to propagate beyond the fixed-statement recovery loop when the difficulty appears to require higher-level intervention.

3.3 Node-Level Routing

When local proving terminates due to budget exhaustion or when evidence of a structural problem emerges, a node reviewer determines how proof construction should proceed. In addition to the local failure history and remaining attempt allowance, the reviewer has access to the parent decomposition, including the assembly proof and sibling statements. This broader context helps distinguish difficulty in proving the current lemma from difficulty caused by the role assigned to that lemma within the proof structure.

The reviewer has three actions. A direct retry is used when local recovery has been exhausted but the current statement and decomposition are still judged appropriate. Unlike local continuation, which further develops the current proving trajectory using primarily local evidence, a direct retry incorporates additional guidance from the node reviewer based on broader proof context, including the parent decomposition and preceding failure history. It initiates a fresh proving attempt on the same statement when higher-level context suggests a concrete alternative without requiring structural revision.

Further decomposition is triggered when the current statement remains appropriate but is difficult to establish directly. The statement is preserved while a set of simpler subgoals is introduced whose verified proofs collectively establish it. The resulting decomposition passes through the same formalization and validation pipeline as described above, preserving the current statement while changing the structure through which it is proved.

Parent-decomposition revision applies when the current node is judged unsuitable for the role assigned to it in the parent decomposition. The parent goal is preserved while the decomposition that introduced the problematic node is replaced. The revision is localized to the affected subgraph, leaving nodes outside this region unchanged. Within the revised decomposition, previously established nodes are retained when the same lemma statement reappears. Nodes that belong only to the discarded decomposition are removed from the active proof graph. However, if such a node has already been proved, its verified result remains available to subsequent proving as an additional verified fact.

Table 1: End-to-end theorem-proving performance on FATE-H, FATE-X, and Putnam 2025. Published results use the inference settings reported by the corresponding works: Mistral AI (2026); Chen et al. (2025); Requena et al. (2026); Chung et al. (2026); Kung et al. (2026); Liu et al. (2026). A dash indicates that a result was not reported.

System	Backbone and Setting	FATE-H	FATE-X	Putnam 2025
Leanstral 1.5	pass@8	87/100	34/100	–
Seed-Prover 1.5	10/40 H20-days/problem[†]	80/100	33/100	11/12
AxProverBase	Claude Opus 4.5, pass@1	66/100	24/100	–
AxProverBase[*]	DeepSeek-V4-Flash, pass@1	28/100	6/100	5/12
Goedel-Architect	DeepSeek-V4-Flash, pass@1[‡]	–	–	11/12
LEAP	Gemini 3.1 Pro, 2 rollouts	–	–	**12**/12
Numina-Lean-Agent	Claude Opus 4.5, variable budget[§]	–	–	**12**/12
OpenMathHarness	DeepSeek-V4-Flash, pass@1	**92**/100	**53**/100	11/12

[†] FATE-H and FATE-X use a maximum budget of 10 H20-days per problem. Putnam 2025 uses a maximum budget of 40 H20-days per problem. [‡] Selected Putnam 2025 problems additionally use natural-language guidance. [§] Numina-Lean-Agent uses approximately $50 per problem by default, with larger budgets for A5 and B6. [*] Reproduced using the released AxProverBase implementation with DeepSeek-V4-Flash. Details are provided in Appendix B.2.

Overall, node-level routing keeps recovery at the current node (direct retry), moves it downward (further decomposition), or propagates it upward (parent-decomposition revision), so that a failure at the current node does not by itself determine how the proof structure changes. By matching the direction and scope of recovery to the source of difficulty, node review provides a mechanism for controlling structural revision.

4 Experiments

4.1 Main Result

We evaluate OMH on FATE-H, FATE-X (Jiang et al., 2026), and Putnam 2025 (Tsoukalas et al., 2024). FATE-H and FATE-X each contain 100 Lean problems at graduate- and PhD-level difficulty respectively. Putnam 2025 contains the twelve problems from the 2025 competition. All OMH results use DeepSeek-V4-Flash (DeepSeek-AI, 2026) with a single end-to-end run per problem. Each run terminates when a verified proof is obtained or the per-problem token budget (100M) is exhausted. Implementation details and inference configurations are provided in Appendix B.1.

Table 1 summarizes the end-to-end results. OMH solves 92/100 problems on FATE-H and 53/100 on FATE-X, with mean inference costs of $0.37 and $1.23 per solved problem respectively. On Putnam 2025, OMH solves 11/12 problems at a mean cost of $0.27 per solved problem. These results show that OMH maintains strong end-to-end performance across benchmarks of substantially different difficulty while keeping inference cost low.

Because published systems use different backbones, rollout counts, and inference budgets, Table 1 reports their published results under the settings used in the corresponding works. For a controlled comparison, we run AxProverBase with DeepSeek-V4-Flash under a 10M token budget. Within the same budget, OMH solves 74/100 FATE-H, 23/100 FATE-X, and 11/12 Putnam 2025 problems, compared with 28/100, 6/100, and 5/12 for AxProverBase. Details can be found in Appendix B.2.

Figure 2 reports, for the single main run of each problem, the cumulative number of problems solved as a function of the tokens consumed before the proof was verified. On FATE-H, OMH reaches 80% of its final solve count within 9.86M tokens and 90% within 20.50M. On FATE-X, the same fractions require 48.09M and 63.59M tokens. OMH continues to convert additional inference budget into verified solutions on FATE-X, where 15 problems are solved only after 30M tokens, compared with four on FATE-H.

4.2 Ablation Study

We evaluate the main design choices of OMH at two complementary levels. First, problem-level ablations measure the contributions of proof decomposition and informal guidance to complete proof

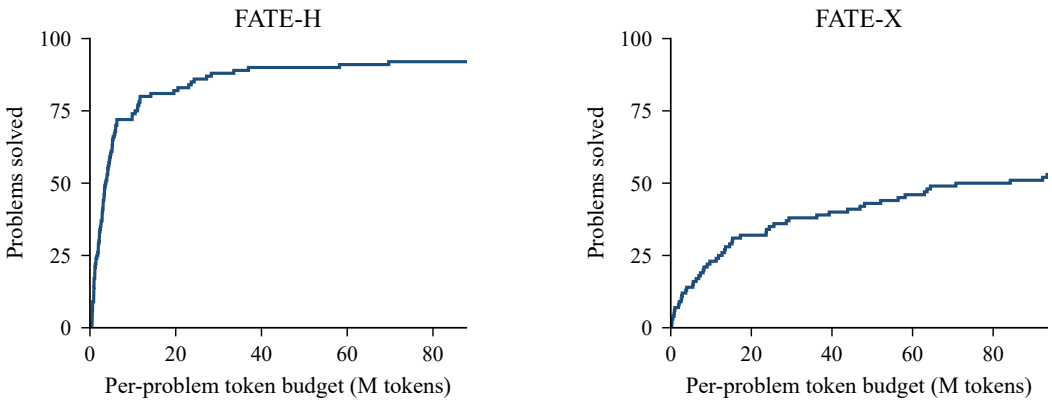

Figure 2: Cumulative number of problems solved within a given per-problem token budget, computed from the single main run of each problem on FATE-H (left) and FATE-X (right). FATE-H saturates early, whereas FATE-X continues to gain verified solutions at larger budgets.

construction. Second, a controlled node-level replay examines the effect of strategy flexibility while holding the node and its preceding context fixed.

Problem-level ablations. We evaluate proof decomposition and informal guidance on a fixed subset of 21 FATE problems, comprising 9 from FATE-H and 12 from FATE-X. We include all problems that OMH solves in the main evaluation using between 15M and 45M tokens. The lower threshold excludes short trajectories that provide limited evidence about the multi-stage proof construction process and the upper threshold keeps the selected problems within a compute range that permits controlled reruns under a common budget. The complete problem list is provided in Appendix B.3. All configurations use DeepSeek-V4-Flash and are evaluated once per problem under a common 25M token ceiling, terminating when either a verified proof is obtained or the budget is exhausted.

We compare full OMH with two ablated variants. *Without Decomposition* disables decomposition throughout the run. No intermediate lemmas are introduced, and decomposition is unavailable as a routing action. *Without Informal Guidance* retains decomposition, formal proving, and review, but removes the generation and revision of informal guidance.

As shown in Table 2, full OMH solves 18 of the 21 problems, compared with 8 without decomposition and 14 without informal guidance. Removing decomposition loses 10 problems solved by OMH and gains none. Every lost problem is solved by full OMH through decomposition, and all failure runs without decomposition exhaust the token budget. These results suggest that decomposition is important when the current goal is too difficult to establish directly. By introducing intermediate lemmas, OMH reduces the difficulty of individual statements and converts a hard proof task into a sequence of tractable subproblems, allowing progress where continued direct proving stalls.

Removing informal guidance loses four problems and gains none, and the four losses share one pattern. On each of them, full OMH preserves the initial one-layer decomposition and closes the corresponding intermediate goals through further formal refinement. The variant without informal guidance further decomposes these goals and fails to complete the resulting proof within the budget, which suggests that guidance is most useful when the current proof structure remains sound but the formal prover has not found an effective strategy. It helps redirect proving within the current goal and avoids unnecessarily moving the recovery to structural level. Thus, the two components play complementary roles: decomposition revises the proof structure when necessary, whereas informal guidance supports continued progress within an appropriate structure.

Node-level replay. We next evaluate the strategy flexibility in the local proving process under controlled local conditions. Because upstream stochasticity can change the decomposition, intermediate nodes, or guidance before entering each node, we isolate this mechanism from upstream variation and evaluate it through node-level replay. We construct an eligible pool of standalone nodes from the OMH main evaluation on FATE-X. Eligible nodes are handled entirely through local proving,

without further decomposition or dependencies on other generated proof-graph nodes. The detailed eligibility criteria are provided in Appendix B.3. We randomly sample 30 nodes, and replay each node from the same recorded local state, fixing its formal statement, proof context, initial informal guidance, and available verified facts. The ablated *Fixed Strategy* variant disables strategy flexibility: the formal prover is instructed through its system prompt to follow the initial informal strategy, while guidance revision is removed from the error reviewer's action space.

OMH solves 28 of the 30 nodes, compared with 25 for *Fixed Strategy*, with all 25 successes of the ablated variant also solved by OMH. On these shared successes, OMH uses an average of 0.98M tokens per node, compared with 1.94M for *Fixed Strategy*. The average number of formal proving rounds similarly increases from 2.04 to 3.28.

An initially reasonable mathematical strategy may be difficult to realize in the given Lean environment. Compiler feedback guides local repairs to a candidate proof but does not lead the prover to reconsider the strategy itself. Strategy flexibility allows repeated formal failures to trigger a revision of the guidance and redirects available proof effort when the current trajectory becomes unproductive. Therefore, it reduces repeated investment in an unfavorable formal trajectory and improves efficiency.

Table 2: Ablations at two levels. Left: problem-level solve counts on 21 FATE problems. Right: node-level replay on 30 FATE-X nodes; token and round means are computed over the 25 nodes solved by both variants.

(a) Problem-level ablation

Variant	**Solved**
Full OMH	18/21
w/o Decomposition	8/21
w/o Informal Guidance	14/21

(b) Node-level replay

Metric	*Full OMH*	*Fixed Strategy*
Solved	28/30	25/30
Mean tokens (M)	0.98	1.94 (1.97×)
Mean rounds	2.04	3.28 (1.61×)

4.3 Review and Routing Analysis

We next examine where OMH spends its recovery effort across different levels of proof construction. We analyze all review and routing events from the 212 main benchmark runs. Among the 2554 local-review verdicts, 92.8% attribute the difficulty to formal proving, while the remaining 7.2% identify issues in the informal guidance or the proof strategy. Local review continues formal proving in 2009 cases (78.7%), revises the informal guidance in 120 (4.7%), and escalates 425 (16.6%) to node-level reconsideration. Of the continuation decisions, 629 subsequently lead to closure of the node within the same local round without revising the guidance. Following guidance revision, 44 cases return to formal proving under the revised strategy, whereas 60 are subsequently escalated to node-level reconsideration when local recovery remains insufficient. The remaining 16 cases are terminated due to budget exhaustion. This pattern shows that OMH generally retains recovery within the current proof trajectory when further local progress remains possible, while progressively broadening the revision scope when narrower interventions are insufficient.

The node reviewer is invoked when the error reviewer selects escalation (425 cases) or when the local proving budget is exhausted before local review (200 cases). Once recovery reaches the node level, the node reviewer uses the surrounding proof context to determine how construction should proceed. Across 625 node-level routing decisions, OMH selects further decomposition in 512 cases, direct retry in 70, and parent-decomposition revision in 43. Of the 70 direct retries, 22 close the node in the subsequent proving attempt. Among the 512 decomposition decisions, 302 produce validated decompositions that are committed to the proof graph, and 133 of these are later fully established and integrated into the proof. These outcomes reveal that the routing distribution has clear priorities that align with the different recovery scopes, while the lower-frequency decisions are followed by successful resolution in a nontrivial subset of the cases in which they are selected.

Parent-decomposition revision operates at the broadest structural scope in OMH, allowing evidence accumulated during local proving to trigger reconsideration of an earlier proof structure. Across the main runs, 16 such revisions are applied across 15 problems. In 7 of these cases, the replacement decomposition is ultimately completed and the revised parent goal is proved. All 7 corresponding runs also prove the root theorem. 14 of the 16 revisions occur after at least one goal in the affected

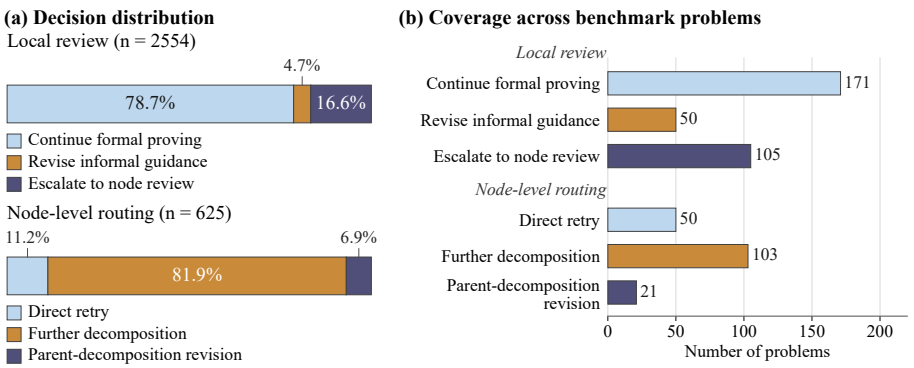

Figure 3: Review and routing behavior across the 212 main benchmark runs. (a) Distribution of 2554 local-review decisions and 625 node-level routing decisions. (b) Number of benchmark problems in which each action occurs at least once. Actions in panel (b) are not mutually exclusive, as a single proof run may invoke multiple recovery mechanisms.

region has already been verified, and 13 previously proved goals are later reused in the revised proof. These cases show that downstream formal evidence can lead OMH to revise an earlier proof plan while continuing to build on useful verified progress. A detailed case study is provided in Appendix C.

To provide an external semantic check of these decisions, we conduct a blinded audit on 50 stratified cases, with 25 local reviews and 25 node-level routing decisions sampled from distinct problems. Two judges from different model families, Claude Opus 4.8 (Anthropic, 2026) and GPT-5.6 Terra (OpenAI, 2026a), evaluate each case using a fixed prompt and only the information available at the corresponding decision point, without access to OMH's selected action or any subsequent proof outcome. Opus 4.8 considers OMH's selected action reasonable in 42/50 cases, including 20/25 local-review and 22/25 node-level decisions, while GPT-5.6 accepts 41/50, including 19/25 and 22/25, respectively. Their preferred actions match OMH's choices in 37/50 and 36/50 cases. These results provide additional evidence that the review and routing behavior observed in complete proof runs is broadly consistent with the information available at the time of each decision. Full sampling details, evaluation protocol, and results are provided in Appendix D.

5 Conclusion

We presented OpenMathHarness (OMH), an agentic theorem-proving framework that keeps mathematical reasoning and formal proving coordinated as a proof develops. Formal outcomes can trigger changes to the current proof strategy, the treatment of a difficult goal, or an earlier decomposition, allowing decisions made at different stages of the proof to be revisited when new evidence emerges. Using DeepSeek-V4-Flash, OMH solves 92/100 problems on FATE-H, 53/100 on FATE-X, and 11/12 on Putnam 2025, with mean inference costs among solved problems ranging from \$0.27 to \$1.23. Controlled experiments examine the effects of key components and run-level analyses characterize how recovery unfolds across different stages of proof construction. Overall, OMH closes the loop between formal proving and proof construction, allowing the proof process to remain coordinated and revisable as formal evidence accumulates.

Limitations. Failure attribution remains inherently ambiguous in some cases, particularly when distinguishing a difficult formal realization from an unsuitable proof strategy. Although OMH provides separate recovery paths for these cases, improving the reliability of this diagnosis remains an important direction. Structural recovery also depends on the quality of generated decompositions, and difficult problems can still require several unsuccessful structural proposals before a useful proof structure is found. Extending this closed-loop proof-construction process to longer, research-scale developments with broader mathematical context is another direction for future work.

AI USE STATEMENT

In this work, we used generative AI tools to provide feedback on research methodology and experimental design; assist with implementation and analysis scripts, interpretation and organization of experimental results, and language refinement during manuscript preparation. This work does not present new mathematical theorems or mathematical proofs as research contributions, so the corresponding disclosure categories are not applicable. The remaining required-disclosure tasks are not applicable to this work.

Additionally, we used generative AI tools for literature retrieval and discovery, code editing, and the presentation of figures and tables. We have reviewed all AI-assisted work. In particular, AI-assisted code and analysis scripts were inspected and tested by the authors, reported quantitative results were checked against the underlying experimental records, and literature identified with AI assistance was verified against the original sources. We take responsibility for the final content of this work, including text, claims, code, and artifacts produced with the aid of generative AI.

REPRODUCIBILITY STATEMENT

We provide an anonymized repository containing the complete implementation of OMH: `https://anonymous.4open.science/r/24793826678a/`. We also provide the same scripts in the supplementary materials. The code will be released publicly after the review process. Additional implementation details, inference settings and configurations are provided in Appendix B. These materials are intended to support reproduction of the reported experiments and analyses.

REFERENCES

Tudor Achim, Alex Best, Alberto Bietti, Kevin Der, Mathïs Fédérico, Sergei Gukov, Daniel Halpern-Leistner, Kirsten Henningsgard, Yury Kudryashov, Alexander Meiburg, Martin Michelsen, Riley Patterson, Eric Rodriguez, Laura Scharff, Vikram Shanker, Vladmir Sicca, Hari Sowrirajan, Aidan Swope, Matyas Tamas, Vlad Tenev, Jonathan Thomm, Harold Williams, and Lawrence Wu. Aristotle: IMO-level automated theorem proving. *arXiv preprint arXiv:2510.01346*, 2025.

Boris Alexeev, Kevin Barreto, Yanyang Li, Jared Duker Lichtman, Liam Price, Jibran Iqbal Shah, Quanyu Tang, and Terence Tao. Primitive sets and von Mangoldt chains: Erdős problem #1196 and beyond. *arXiv preprint arXiv:2605.00301*, 2026.

Anthropic. Introducing Claude Opus 4.8, 2026. URL `https://www.anthropic.com/news/claude-opus-4-8`. Accessed: Sep, 2026.

Zhangir Azerbayev, Hailey Schoelkopf, Keiran Paster, Marco Dos Santos, Stephen Marcus McAleer, Albert Q. Jiang, Jia Deng, Stella Biderman, and Sean Welleck. Llemma: An open language model for mathematics. In *The Twelfth International Conference on Learning Representations*, 2024. URL `https://openreview.net/forum?id=4WnqRR915j`.

Kaito Baba, Chaoran Liu, Shuhei Kurita, and Akiyoshi Sannai. Prover Agent: An agent-based framework for formal mathematical proofs. *arXiv preprint arXiv:2506.19923*, 2025.

Jiangjie Chen, Wenxiang Chen, Jiacheng Du, Jinyi Hu, Zhicheng Jiang, Allan Jie, Xiaoran Jin, Xing Jin, Chenggang Li, Wenlei Shi, Zhihong Wang, Mingxuan Wang, Chenrui Wei, Shufa Wei, Huajian Xin, Fan Yang, Weihao Gao, Zheng Yuan, Tianyang Zhan, Zeyu Zheng, Tianxi Zhou, and Thomas Hanwen Zhu. Seed-Prover 1.5: Mastering undergraduate-level theorem proving via learning from experience. *arXiv preprint arXiv:2512.17260*, 2025.

Jui-Hui Chung, Ziyang Cai, Zihao Li, Qishuo Yin, Rohit Agarwal, Simon Park, Rodrigo Porto, Narutatsu Ri, Ziran Yang, Shange Tang, Xingyu Dang, Hongzhou Lin, Mengdi Wang, Danqi Chen, Chi Jin, Liam H Fowl, and Sanjeev Arora. Goedel-Architect: Streamlining formal theorem proving with blueprint generation and refinement. *arXiv preprint arXiv:2606.06468*, 2026.

Leonardo de Moura and Sebastian Ullrich. The Lean 4 theorem prover and programming language. In *International Conference on Automated Deduction*, pp. 625–635. Springer, 2021.

DeepSeek-AI. DeepSeek-V4: Towards highly efficient million-token context intelligence. *arXiv preprint arXiv:2606.19348*, 2026.

Marco Dos Santos, Hugues de Saxcé, Haiming Wang, Ran Wang, Mantas Baksys, Mert Unsal, Junqi Liu, Zhengying Liu, and Jia Li. Kimina Lean Server: A high-performance Lean server for large-scale verification. *arXiv preprint arXiv:2504.21230*, 2025.

Tony Feng, Trieu Trinh, Garrett Bingham, Jiwon Kang, Shengtong Zhang, Sang hyun Kim, Kevin Barreto, Carl Schildkraut, Junehyuk Jung, Jaehyeon Seo, Carlo Pagano, Yuri Chervonyi, Dawsen Hwang, Kaiying Hou, Sergei Gukov, Cheng-Chiang Tsai, Hyunwoo Choi, Youngbeom Jin, Wei-Yuan Li, Hao-An Wu, Ruey-An Shiu, Yu-Sheng Shih, Quoc V. Le, and Thang Luong. Semi-autonomous mathematics discovery with Gemini: A case study on the Erdős problems. *arXiv preprint arXiv:2601.22401*, 2026.

Moritz Firsching, Paul Lezeau, Salvatore Mercuri, Miklós Z. Horváth, Yaël Dillies, Calle Sönne, Eric Wieser, Fred Zhang, Thomas Hubert, Blaise Agüera y Arcas, and Pushmeet Kohli. Formal Conjectures: An open and evolving benchmark for verified discovery in mathematics. *arXiv preprint arXiv:2605.13171*, 2026.

Guoxiong Gao, Zeming Sun, Jiedong Jiang, Yutong Wang, Jingda Xu, Peihao Wu, Bryan Dai, and Bin Dong. LeanSearch V2: Global premise retrieval for Lean 4 theorem proving. *arXiv preprint arXiv:2605.13137*, 2026.

Elliot Glazer, Ege Erdil, Tamay Besiroglu, Diego Chicharro, Evan Chen, Alex Gunning, Caroline Falkman Olsson, Jean-Stanislas Denain, Anson Ho, Emily de Oliveira Santos, Olli

Järviniemi, Matthew Barnett, Robert Sandler, Matej Vrzala, Jaime Sevilla, Qiuyu Ren, Elizabeth Pratt, Lionel Levine, Grant Barkley, Natalie Stewart, Bogdan Grechuk, Tetiana Grechuk, Shreepranav Varma Enugandla, and Mark Wildon. FrontierMath: A benchmark for evaluating advanced mathematical reasoning in AI. *arXiv preprint arXiv:2411.04872*, 2024.

Yichen Huang and Lin F Yang. Winning gold at IMO 2025 with a model-agnostic verification-and-refinement pipeline. *arXiv preprint arXiv:2507.15855*, 2025.

Albert Qiaochu Jiang, Sean Welleck, Jin Peng Zhou, Timothee Lacroix, Jiacheng Liu, Wenda Li, Mateja Jamnik, Guillaume Lample, and Yuhuai Wu. Draft, Sketch, and Prove: Guiding formal theorem provers with informal proofs. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=SMa9EAovKMC.

Jiedong Jiang, Wanyi He, Yuefeng Wang, Guoxiong Gao, Yongle Hu, Jingting Wang, Nailin Guan, Peihao Wu, Bryan Dai, Liang Xiao, and Bin Dong. FATE: A formal benchmark series for frontier algebra of multiple difficulty levels. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=3bD19r4jqh.

Po-Nien Kung, Linfeng Song, Dawsen Hwang, Jinsung Yoon, Chun-Liang Li, Simone Severini, Mirek Olšák, Edward Lockhart, Quoc V Le, Burak Gokturk, Thang Luong, Tomas Pfister, and Nanyun Peng. LEAP: Supercharging LLMs for formal mathematics with agentic frameworks. *arXiv preprint arXiv:2606.03303*, 2026.

Lean Prover Community. Lean REPL: A read-eval-print loop for Lean 4. https://github.com/leanprover-community/repl, 2026. Accessed: Sep, 2026.

Haohan Lin, Zhiqing Sun, Sean Welleck, and Yiming Yang. Lean-STaR: Learning to interleave thinking and proving. In *The Thirteenth International Conference on Learning Representations*, 2025a. URL https://openreview.net/forum?id=SOWZ59UyNc.

Yong Lin, Shange Tang, Bohan Lyu, Jiayun Wu, Hongzhou Lin, Kaiyu Yang, Jia Li, Mengzhou Xia, Danqi Chen, Sanjeev Arora, and Chi Jin. Goedel-Prover: A frontier model for open-source automated theorem proving. In *Second Conference on Language Modeling*, 2025b. URL https://openreview.net/forum?id=x2y9i2HDjD.

Yong Lin, Shange Tang, Bohan Lyu, Ziran Yang, Jui-Hui Chung, Haoyu Zhao, Lai Jiang, Yihan Geng, Jiawei Ge, Jingruo Sun, Jiayun Wu, Jiri Gesi, Ximing Lu, David Acuna, Kaiyu Yang, Hongzhou Lin, Yejin Choi, Danqi Chen, Sanjeev Arora, and Chi Jin. Goedel-Prover-V2: Scaling formal theorem proving with scaffolded data synthesis and self-correction. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=j4C0nALrgK.

Junqi Liu, Zihao Zhou, Zekai Zhu, Marco Dos Santos, Weikun He, Jiawei Liu, Yunzhou Xie, Junqiao Zhao, Qiufeng Wang, Lihong Zhi, Jia Li, and Wenda Li. Numina-Lean-Agent: An open and general agentic reasoning system for formal mathematics. In *Forty-third International Conference on Machine Learning*, 2026. URL https://openreview.net/forum?id=0bTEd4LpQr.

Thang Luong, Dawsen Hwang, Hoang H. Nguyen, Golnaz Ghiasi, Yuri Chervonyi, Insuk Seo, Junsu Kim, Garrett Bingham, Jonathan Lee, Swaroop Mishra, Alex Zhai, Clara Huiyi Hu, Henryk Michalewski, Jimin Kim, Jeonghyun Ahn, Junhwi Bae, Xingyou Song, Trieu H. Trinh, Quoc V. Le, and Junehyuk Jung. Towards robust mathematical reasoning. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pp. 35406–35430, 2025.

Mistral AI. Leanstral 1.5: Proof abundance for all. https://mistral.ai/news/leanstral-1-5/, 2026. Accessed: Sep, 2026.

OpenAI. GPT-5.6: Frontier intelligence that scales with your ambition, 2026a. URL https://openai.com/index/gpt-5-6/. Accessed: Sep, 2026.

OpenAI. Planar point sets with many unit distances. https://cdn.openai.com/pdf/74c24085-19b0-4534-9c90-465b8e29ad73/unit-distance-proof.pdf, 2026b. Accessed: Sep, 2026.

Azim Ospanov, Farzan Farnia, and Roozbeh Yousefzadeh. APOLLO: Automated LLM and Lean collaboration for advanced formal reasoning. *Advances in Neural Information Processing Systems*, 38:41599–41633, 2025.

Ruichen Qiu, Yichuan Cao, Junqi Liu, Dakai Guo, Xiao-Shan Gao, Lihong Zhi, and Ruyong Feng. Mechanic: Sorrifier-driven formal decomposition workflow for automated theorem proving. *arXiv preprint arXiv:2603.24465*, 2026.

Z.Z. Ren, Zhihong Shao, Junxiao Song, Huajian Xin, Haocheng Wang, Wanjia Zhao, Liyue Zhang, Zhe Fu, Qihao Zhu, Dejian Yang, Z.F. Wu, Zhibin Gou, Shirong Ma, Hongxuan Tang, Yuxuan Liu, Wenjun Gao, Daya Guo, and Chong Ruan. DeepSeek-Prover-V2: Advancing formal mathematical reasoning via reinforcement learning for subgoal decomposition. *arXiv preprint arXiv:2504.21801*, 2025.

Borja Requena, Austin Letson, Krystian Nowakowski, Izan Beltran Ferreiro, and Leopoldo Sarra. A minimal agent for automated theorem proving. In *Forty-third International Conference on Machine Learning*, 2026. URL https://openreview.net/forum?id=QGAOswfVvD.

Amitayush Thakur, George Tsoukalas, Yeming Wen, Jimmy Xin, and Swarat Chaudhuri. An in-context learning agent for formal theorem-proving. In *First Conference on Language Modeling*, 2024. URL https://openreview.net/forum?id=V7HRrxXUhN.

The mathlib Community. The Lean mathematical library. In *Proceedings of the 9th ACM SIGPLAN International Conference on Certified Programs and Proofs*, CPP 2020, pp. 367–381, New York, NY, USA, 2020. Association for Computing Machinery. ISBN 9781450370974.

George Tsoukalas, Jasper Lee, John Jennings, Jimmy Xin, Michelle Ding, Michael Jennings, Amitayush Thakur, and Swarat Chaudhuri. PutnamBench: Evaluating neural theorem-provers on the Putnam mathematical competition. *Advances in Neural Information Processing Systems*, 37: 11545–11569, 2024.

Sumanth Varambally, Thomas Voice, Yanchao Sun, Zhifeng Chen, Rose Yu, and Ke Ye. Hilbert: Recursively building formal proofs with informal reasoning. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=GN8OdkTo3B.

Haiming Wang, Huajian Xin, Chuanyang Zheng, Zhengying Liu, Qingxing Cao, Yinya Huang, Jing Xiong, Han Shi, Enze Xie, Jian Yin, Zhenguo Li, and Xiaodan Liang. LEGO-Prover: Neural theorem proving with growing libraries. In *The Twelfth International Conference on Learning Representations*, 2024a. URL https://openreview.net/forum?id=3f5PALef5B.

Haiming Wang, Mert Unsal, Xiaohan Lin, Mantas Baksys, Junqi Liu, Marco Dos Santos, Flood Sung, Marina Vinyes, Zhenzhe Ying, Zekai Zhu, Jianqiao Lu, Hugues de Saxcé, Bolton Bailey, Chendong Song, Chenjun Xiao, Dehao Zhang, Ebony Zhang, Frederick Pu, Han Zhu, Jiawei Liu, Jonas Bayer, Julien Michel, Longhui Yu, Léo Dreyfus-Schmidt, Lewis Tunstall, Luigi Pagani, Moreira Machado, Pauline Bourigault, Ran Wang, Stanislas Polu, Thibaut Barroyer, Wen-Ding Li, Yazhe Niu, Yann Fleureau, Yangyang Hu, Zhouliang Yu, Zihan Wang, Zhilin Yang, Zhengying Liu, and Jia Li. Kimina-Prover Preview: Towards large formal reasoning models with reinforcement learning. *arXiv preprint arXiv:2504.11354*, 2025.

Jianing Wang, Jianfei Zhang, Qi Guo, Linsen Guo, Rumei Li, Chao Zhang, Chong Peng, Cunguang Wang, Dengchang Zhao, Jiarong Shi, Jingang Wang, Liulin Feng, Mengxia Shen, Qi Li, Shengnan An, Shun Wang, Wei Shi, Xiangyu Xi, Xiaoyu Li, Xuezhi Cao, Yi Lu, Yunke Zhao, Zhengyu Chen, Zhimin Lin, Wei Wang, Peng Pei, and Xunliang Cai. LongCat-Flash-Prover: Advancing native formal reasoning via agentic tool-integrated reinforcement learning. *arXiv preprint arXiv:2603.21065*, 2026.

Ruida Wang, Jipeng Zhang, Yizhen Jia, Rui Pan, Shizhe Diao, Renjie Pi, and Tong Zhang. Theoremllama: Transforming general-purpose LLMs into Lean4 experts. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pp. 11953–11974, 2024b.

Huajian Xin, Z.Z. Ren, Junxiao Song, Zhihong Shao, Wanjia Zhao, Haocheng Wang, Bo Liu, Liyue Zhang, Xuan Lu, Qiushi Du, Wenjun Gao, Haowei Zhang, Qihao Zhu, Dejian Yang, Zhibin Gou, Z.F. Wu, Fuli Luo, and Chong Ruan. DeepSeek-Prover-V1.5: Harnessing proof assistant feedback for reinforcement learning and Monte-Carlo tree search. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=I4YAIwrsXa.

Youyuan Zhang, Jialiang Sun, Hangrui Bi, Chuqin Geng, Wenjie Ma, Zhaoyu Li, and Xujie Si. DreamProver: Evolving transferable lemma libraries via a wake-sleep theorem-proving agent. *arXiv preprint arXiv:2604.26311*, 2026.

Kunhao Zheng, Jesse Michael Han, and Stanislas Polu. miniF2F: a cross-system benchmark for formal olympiad-level mathematics. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=9ZPegFuFTFv.

Yichi Zhou, Jianqiu Zhao, Yongxin Zhang, Bohan Wang, Siran Wang, Luoxin Chen, Jiahui Wang, Haowei Chen, Allan Jie, Xinbo Zhang, Haocheng Wang, Luong Trung, Rong Ye, Phan Nhat Hoang, Huishuai Zhang, Peng Sun, and Hang Li. Solving formal math problems by decomposition and iterative reflection. *arXiv preprint arXiv:2507.15225*, 2025.

APPENDIX

A ALGORITHM

Algorithm 1 summarizes the proof construction process of OMH. PLAN, DECOMPOSE, FORMALIZEANDVALIDATE, and INFORMALGUIDANCE each denote a bounded propose-review loop. FORMALPROVE denotes a bounded formal proving loop with iterative Lean feedback. All stage budgets and routing bounds follow Table 5.

Algorithm 1 Proof construction in OMH

Require: Target theorem T
Ensure: A verified proof of T, or UNRESOLVED

$P \leftarrow \text{PLAN}(T)$; $G \leftarrow$ proof graph rooted at T
if P proposes a decomposition D **then**
  $D \leftarrow \text{FORMALIZEANDVALIDATE}(D)$
  if D is accepted **then**
    $\text{COMMITDECOMPOSITION}(G, D)$
  end if
end if
while T is not proved and the problem budget remains **do**
  $v \leftarrow \text{SELECTUNRESOLVEDNODE}(G)$
  $g \leftarrow \text{INFORMALGUIDANCE}(v, G)$
  while v is unresolved and local recovery remains available **do**
    $\pi \leftarrow \text{FORMALPROVE}(v, g, G)$
    if $\text{VERIFY}(v, \pi)$ **then**
      $\text{COMMITPROOF}(G, v, \pi)$; $\text{PROPAGATEVERIFIEDPROGRESS}(G)$
      break
    end if
    $a \leftarrow \text{ERRORREVIEW}(v, g, \pi)$
    if $a =$ CONTINUEFORMALPROVING **then**
      continue the current formal trajectory
    else if $a =$ REVISEINFORMALGUIDANCE **then**
      $g \leftarrow \text{INFORMALGUIDANCE}(v, G)$
      restart a fresh formal-proving loop
    else if $a =$ ESCALATETONODEREVIEW **then**
      break
    end if
  end while
  if v is proved **then**
    continue
  end if
  $a \leftarrow \text{NODEREVIEW}(v, G)$
  if $a =$ DIRECTRETRY **then**
    start a fresh local attempt on the same statement
  else if $a =$ FURTHERDECOMPOSITION **then**
    $D \leftarrow \text{DECOMPOSE}(v, G)$
    $D \leftarrow \text{FORMALIZEANDVALIDATE}(D)$
    if D is accepted **then**
      $\text{COMMITDECOMPOSITION}(G, D)$
    end if
  else if $a =$ PARENTDECOMPOSITIONREVISION **then**
    $D \leftarrow \text{REVISEPARENTDECOMPOSITION}(v, G)$
    $D \leftarrow \text{FORMALIZEANDVALIDATE}(D)$
    if D is accepted **then**
      $\text{REPLACEDECOMPOSITION}(G, D)$
    end if
  end if
end while
return $\text{FINALVERIFY}(G, T)$ if T is proved; otherwise UNRESOLVED

B EXPERIMENTAL DETAILS

This appendix provides the experimental and implementation details needed to interpret and reproduce the results in the main paper. We first describe the implementation and inference configuration used by OMH, the settings of the reproduced and externally reported baselines, and the protocols used for the two ablation studies, then report an additional evaluation on FC100SolvedSet. The prompt templates, configuration files, and scripts of OMH are included in the supplementary materials.

B.1 IMPLEMENTATION SETTINGS

Backbone model and cost accounting. All components of OMH use DeepSeek-V4-Flash as the backbone model, including proof decomposition, informal reasoning, formal proving, and all levels of review. All reported runs use the model with thinking mode enabled at high reasoning effort. Under such setting, temperature has no effect and top-p is 1.0 by default. OMH uses eight parallel workers for dispatched proof nodes.

Inference cost is computed from the token usage of every DeepSeek-V4-Flash request using the official DeepSeek pricing applicable to our experiments. For a request r, we record the numbers of cache-miss input tokens $n^{(r)}_{\text{miss}}$, cache-hit input tokens $n^{(r)}_{\text{hit}}$, and output tokens $n^{(r)}_{\text{out}}$. Reasoning tokens generated in thinking mode are included in the reported output-token count and are therefore included in all cost calculations. The cost of a request is computed by

$$c_r = p_{\text{miss}} n^{(r)}_{\text{miss}} + p_{\text{hit}} n^{(r)}_{\text{hit}} + p_{\text{out}} n^{(r)}_{\text{out}},$$

where $p_{\text{miss}} = 0.14 \times 10^{-6}$, $p_{\text{hit}} = 0.0028 \times 10^{-6}$, and $p_{\text{out}} = 0.28 \times 10^{-6}$ are the corresponding official DeepSeek prices. The cost of a problem is the sum of these request-level costs over its complete end-to-end run. We report two cost measures. The mean cost per successful run: \$0.37 on FATE-H, \$1.23 on FATE-X, and \$0.27 on Putnam 2025. The mean cost per attempted problem includes unsolved runs: \$0.49 on FATE-H and \$1.62 on FATE-X.

Formal environment and verification. All three main benchmarks are evaluated under Lean 4.26.0 in their corresponding benchmark project environments, with the Mathlib version and local dependencies fixed by each benchmark project. OMH uses Lean compilation both for interactive proof development and for proof verification. The implementation supports two interchangeable execution backends, a Lean REPL (Lean Prover Community, 2026) and a Kimina Lean server (Dos Santos et al., 2025). The choice of backend does not affect the verification outcome for a given formal proof under the same environment.

Lean is used for three distinct purposes in OMH. First, during formal proving, candidate proofs are compiled iteratively and the resulting diagnostics are returned to the formal prover. These intermediate compilations provide search feedback and are not treated as proof acceptance. Second, a proposed decomposition is structurally validated in Lean before it is accepted. During the check, the proposed child lemmas are introduced with `sorry` bodies. The parent assembly proof itself must be complete and contain no proof placeholders. Successful compilation verifies that the child statements are well typed and that the child statements are collectively sufficient to derive the parent goal, conditional on the child lemmas being proved separately. It does not count any child lemma as proved. Finally, a node is marked as proved only after a complete candidate proof passes verification in the corresponding benchmark environment. Once all required children of a decomposition have been verified, the parent assembly is checked again with the verified child proofs before the parent is accepted.

Before a problem is counted as solved, OMH reconstructs the final Lean file from the original benchmark context and theorem statement together with the verified declarations produced during proof construction. The reconstructed file is compiled from scratch and must contain no unresolved proof placeholders or disallowed constructs that introduce unverified assumptions. We further inspect the transitive axiom dependencies of the root theorem using `#print axioms` and require them to be contained in {`propext`, `Classical.choice`, `Quot.sound`}. A run is accepted only if all final verification checks succeed.

Tool configuration. OMH exposes a small set of tools for interacting with the Lean environment and with information accumulated during proof construction. Table 3 summarizes the tools used in the reported experiments. In the experiments, OMH has no additional web search tools except for LeanSearch and uses no human-provided hints. Experiences shared by Read Proof and Read Skill tools are summarized from nodes established within the current run and are not shared across problems.

Table 3: Tools available to OMH during proof construction.

Tool	Function
LeanSearch	Retrieves relevant Lean and Mathlib declarations from a natural-language description of the mathematical result required by the current proof.
Lean Compile	Compiles a candidate Lean fragment in the current environment and returns compiler diagnostics for iterative proof development. Intermediate compilation is only used as feedback and is distinct from proof acceptance.
Read Proof	Retrieves the verified proof of an already established node in the proof graph when its proof structure is useful for the current task.
Read Skill	Retrieves a selected reusable proving strategy summarized from the already established nodes.

Role configuration. Table 4 summarizes the information available to each model role and its output, while Table 5 reports the inference budgets and routing bounds used in the main experiments.

Table 4: Configuration of the model roles used in the main OMH experiments.

Stage	Context → Output
Planning	Root theorem, local definitions, and elaborated goal → initial and reviewed proof plan.
Decomposition	Current goal, verified facts, current proof structure, and previous decompositions → child lemmas, parent assembly, and reviewed decomposition.
Formalization	Reviewed decomposition, informal lemma descriptions, and Lean context → formal child statements, formal parent assembly, and faithfulness judgment.
Informal guidance	Current goal, verified facts, proof-plan context, and previous guidance or feedback → reviewed informal proof strategy.
Formal proving	Fixed Lean statement, informal guidance, verified facts, compiler diagnostics, and retrieved information → candidate Lean proof.
Local review	Current statement, informal guidance, latest formal attempt, compiler diagnostics, and local review history → decision and diagnosis for continued formal proving, informal guidance revision or escalation to node review.
Node reviewer	Current goal, local failure history, parent decomposition, sibling state, and remaining allowance → decision and diagnosis for direct retry, further decomposition, or parent-decomposition revision.

Table 5: Inference budgets and routing bounds used in the main experiments.

Stage	Budget / bound
Planning	1.0M tokens
Decomposition	1.2M tokens
Formalization	5.0M tokens
Informal guidance	1.8M tokens
Formal proving	1.8M tokens
Local review	3 returns per attempt; at most 1 to informal guidance
Parent-decomposition revision	2 per attempt; 6 per problem

B.2 Main Evaluation Details

Baseline settings and controlled comparison. We additionally evaluate AxProverBase using DeepSeek-V4-Flash to provide a same-backbone comparison with OMH. We use the released Ax-

ProverBase implementation and replace its backbone model with DeepSeek-V4-Flash for all components. Web search is disabled. Each problem is limited to 20 proposer iterations and 10M cumulative model tokens. Candidate proofs are verified in the same Lean 4.26.0 benchmark environments as OMH and are subjected to the same final verification and axiom-audit criterion.

Table 6: Same-backbone comparison between OMH and AxProverBase at a common 10M token budget. Both systems use DeepSeek-V4-Flash as backbone.

System	FATE-H	FATE-X	Putnam 2025
OpenMathHarness	74/100	23/100	11/12
AxProverBase (reproduced)	28/100	6/100	5/12

To provide a more controlled comparison with OMH, Table 6 also reports the number of OMH problems solved under 10M token budget setting. OMH solves 74/100 FATE-H, 23/100 FATE-X, and 11/12 Putnam 2025 problems, compared with 28/100, 6/100, and 5/12 for the reproduced AxProverBase.

Putnam 2025 per-problem statistics. On Putnam 2025, OMH solves 11/12 problems at an average cost of $0.27 per solved problem. Table 7 reports the LLM effort, proof-graph size, and final Lean proof length for each Putnam 2025 problem. In comparison, Seed-Prover 1.5 and Goedel-Architect (with NL) also report 11/12. LEAP and Numina-Lean-Agent report 12/12 under substantially different inference regimes, including multiple rollouts and auxiliary natural-language guidance. A5, the only problem unsolved by OMH, is also the most compute-intensive instance in LEAP. Numina-Lean-Agent solves it using a dedicated subagent and a substantially enlarged budget. We therefore report these results under their native inference settings.

Table 7: Per-problem OMH results on Putnam 2025.

Problem	A1	A2	A3	A4	A6	B1	B2	B3	B4	B5	B6
LLM calls	146	84	123	54	81	67	134	73	94	137	128
Proof-graph nodes	6	5	5	3	4	3	5	4	4	5	6
Code length (lines)	479	765	713	443	368	404	1376	496	576	1287	551

B.3 Ablation Details

The 21 problems used in the problem-level ablation are listed in Table 8.

Table 8: Problems used in the problem-level ablation.

Benchmark	Problems selected
FATE-H	023, 031, 032, 045, 052, 055, 059, 060, 095
FATE-X	006, 007, 008, 011, 016, 020, 034, 036, 050, 057, 068, 089

For the node-level replay, we first construct an eligible pool of standalone proof nodes from the FATE-X main runs. A node is eligible if it is attempted directly without further decomposition, has no dependency on other generated proof-graph nodes, and its proof does not reference sibling lemmas. Eligibility does not depend on the outcome of the original problem. Nodes from both solved and unresolved problems are included. We then randomly sample 30 nodes from this pool for replay. Each replay starts from the recorded state at the beginning of local formal proving, before any local-review decision or guidance revision.

B.4 Additional Evaluation Results

We additionally evaluate OMH on FC100SolvedSet, an official frozen evaluation subset of the Formal Conjectures benchmark (Firsching et al., 2026), consisting of 100 statements including solved research problems, test statements, API statements, and textbook problems.

Each target is converted to the single-theorem interface used by OMH. Existing proof content and unrelated sibling theorems are removed from the model input. The definitions, notation, instances,

Table 9: Additional evaluation on FC100SolvedSet. The five pre-inference exclusions are not counted as theorem-proving failures.

Category	Problems	Evaluated	Solved
Test statement	34	33	33
API statement	9	9	9
Textbook problem	7	6	6
Solved research problem	50	47	16
All	100	95	64

and other formal context required to state and verify the target are retained. The resulting files are checked to preserve the original target statements and to contain no accessible proof of the target.

Five problems are excluded before evaluation because they cannot be isolated into self-contained proof tasks under our verification policy. Four depend on upstream declarations with `sorry` placeholder proofs. The other one depends on a kernel-opaque partial definition whose available proof requires `native_decide` disallowed by our verifier. We report these exclusions separately from theorem-proving failures, leaving 95 evaluated problems.

We use DeepSeek-V4-Flash as the backbone LLM. Evaluation is performed in the Formal Conjectures benchmark environment with Lean 4.27.0 and its fixed Mathlib revision. Each problem receives one scored end-to-end run under a 40M token budget. A problem is counted as solved only when the assembled proof passes final Lean verification and the axiom audit.

As shown in Table 9, OMH proves 64 of the 95 evaluated problems. It solves all 48 evaluated test, API, and textbook statements and 16 of the 47 evaluated solved-research problems. The clear gap across categories indicates that research-level problems remain substantially more challenging. OMH extends the evaluation beyond the competition and algebra-focused main benchmarks while also leaving substantial room for improvement on research-level theorem proving.

C Case Study

We use FATE-H 95 to illustrate how the components of OMH interact within a single end-to-end proof trajectory. The theorem states that if $x_1, \ldots, x_r$ is an M-regular sequence in a commutative ring R, then $x_1^{a_1}, \ldots, x_r^{a_r}$ is again M-regular for positive integers $a_1, \ldots, a_r$. The initial proof plan decomposes the theorem into three first-layer lemmas following an induction on the regular sequence. L_1 shows that powers preserve regularity of a single element; L_2 transfers regularity of the tail from the quotient by x to the quotient by x^k; and L_3 formulates the powered-sequence result over an arbitrary module, providing the inductive step through quotient modules. The decomposition and its parent assembly pass formalization and validation and are committed to the proof graph.

The three first-layer lemmas are dispatched in parallel and follow different proving trajectories. For L_1, the formal prover identifies an existing library theorem establishing the required result, and the node is verified on the second attempt. L_2 and L_3, by contrast, remain unresolved after repeated formal refinement. Their failures continue to appear compatible with mathematically sound statements and strategies, so the error reviewer initially keeps both nodes within local formal recovery. Once further local refinement ceases to make progress, both nodes are escalated to node-level review. In each case, the node reviewer judges the current statement to remain appropriate but the direct formal task to be too difficult, and routes the node to further decomposition.

Further decomposition resolves L_2 by separating the missing exact-sequence machinery from the regularity argument itself. The new decomposition introduces $L_{2.1}$, which constructs the short exact sequence connecting consecutive power quotients, and $L_{2.2}$, which transfers regularity from the two end terms of such a sequence to the middle term. The first formalization of $L_{2.1}$ passes structural checking but does not faithfully preserve the proposed lemma. It asserts only the existence of linear maps forming an exact sequence, whereas the informal decomposition specifies the canonical quotient maps required by the argument. The faithfulness reviewer rejects this formalization, and the defining equations for the quotient maps are restored in the revision. Both child lemmas are subsequently proved, and rechecking the assembly against their verified proofs establishes L_2.

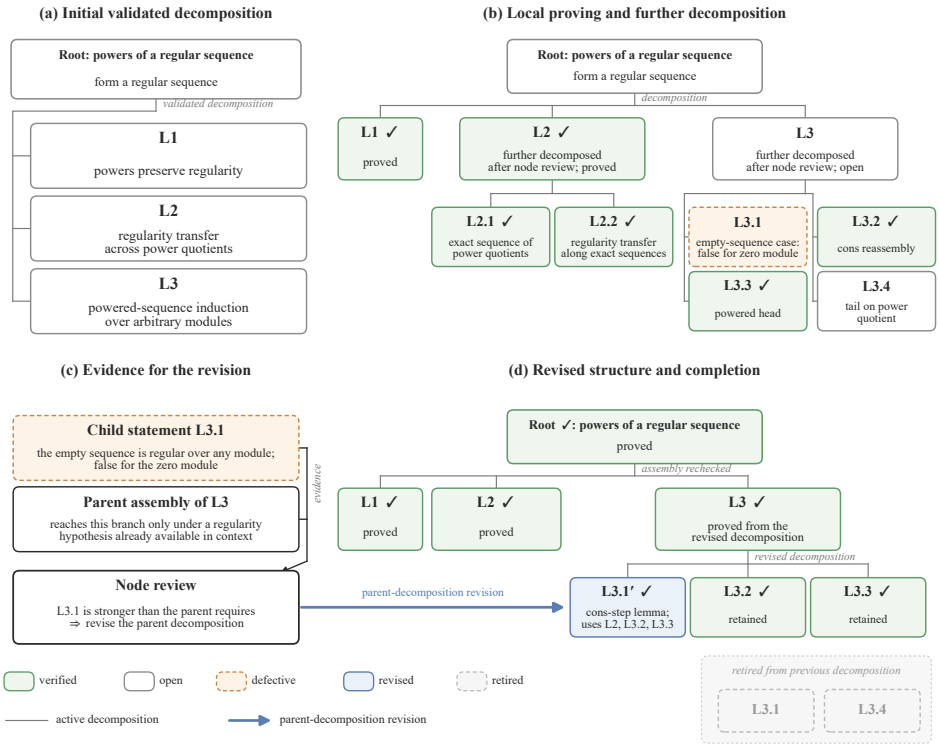

Figure 4: Proof trajectory for FATE-H 95. (a) OMH begins from a validated decomposition into three first-layer lemmas. (b) L_1 is proved directly. L_2 and L_3 are further decomposed after node-level review. The resulting subgoals establish L_2 and partially resolve L_3, where $L_{3.1}$ is identified as defective. (c) The statement of $L_{3.1}$ is false for the zero module while the parent assembly requires this branch only under a regularity hypothesis already available in context. Node review therefore requests revision of the parent decomposition. (d) The revised decomposition replaces the defective branch with $L'_{3.1}$, retains the verified $L_{3.2}$ and $L_{3.3}$, and jointly uses these lemmas to establish L_3 and complete the root theorem.

The decomposition of L_3 exposes a more consequential failure. The node reviewer separates the induction into an empty-list case $L_{3.1}$, a cons-case reassembly $L_{3.2}$, regularity of the powered head $L_{3.3}$, and regularity of the tail over the quotient by a power $L_{3.4}$. Two of these lemmas are proved quickly, where $L_{3.2}$ is verified directly and $L_{3.3}$ is proved by reusing the already verified L_1. The difficulty concentrates on $L_{3.1}$, which asserts that the empty sequence is regular over an arbitrary module. Formal proving repeatedly fails, and the prover reports that the statement lacks the non-triviality condition required by the corresponding definition. After repeated failures, the prover identifies a counterexample using the zero module and concludes that $L_{3.1}$ is false as stated. The local recovery is immediately terminated, forwarding the defective goal to node-level review.

At node level, the surrounding proof structure provides the evidence needed to determine the appropriate scope of revision. Inspecting the decomposition of L_3, the node reviewer observes that the parent induction never requires the unconditional empty-list statement represented by $L_{3.1}$. The relevant branch is reached only under the regularity hypothesis already carried by the parent proof. The failed child lemma has been stated more generally than the assembly requires. The reviewer therefore requests revision of the parent decomposition. The defective $L_{3.1}$ and the unresolved $L_{3.4}$ are removed from the active structure while the verified $L_{3.2}$ and $L_{3.3}$ are retained. The revised decomposition introduces only one new cons-step lemma $L'_{3.1}$, which is proved on its first formal attempt by combining L_2, $L_{3.2}$, and $L_{3.3}$. Rechecking the revised assembly establishes L_3, after which the root assembly is rechecked using the verified L_1, L_2, and L_3, completing the theorem.

FATE-H 95 illustrates the diagnosis-driven revision underlying OMH. Formal failure does not lead to a fixed recovery action. When the current objective remains appropriate, as in L_2, the statement is preserved and additional proof structure is introduced. When the accumulated evidence reveals a

defect in an intermediate objective, as in $L_{3.1}$, reconsideration is propagated to the parent decomposition. In both cases, the scope of revision is determined by the diagnosed source of difficulty, while verified results outside the affected region are retained and reused. The case therefore shows how downstream formal evidence can reshape earlier proof decisions without discarding valid progress, closing the feedback loop between formal proving and proof construction.

D External Audit of Review and Routing Decisions

We conduct a blinded external-model audit to provide an additional semantic consistency check of the review and routing behavior analyzed in Section 4.3. The audit evaluates whether OMH's review and routing decisions are consistent with the mathematical and formal evidence available at each decision point.

The audit contains 50 decision points from distinct benchmark problems, with 25 local-review decisions and 25 node-level routing decisions. Sampling is stratified over the six actions in Table 10 to ensure representation of less frequent actions. At most one decision is sampled from each problem. Local cases are selected from decisions with complete recorded review contexts, and node-level cases use the first review call in each routing episode. Sampling follows a fixed procedure without reference to subsequent proof outcomes.

To evaluate each decision in its original context, we present the auditors with the proof state and recovery history available to OMH at that point. This includes the current goal and guidance, the latest proof attempt and feedback from Lean server, previously verified facts, and the remaining recovery budget. For node-level decisions, we also provide the surrounding decomposition and sibling states when available. The auditors receive only the decision-time context, with OMH's selected action, the corresponding attribution label, and all subsequent events and outcomes withheld.

Two judges from different model families, Claude Opus 4.8 (Anthropic, 2026) and GPT-5.6 Terra (OpenAI, 2026a), independently evaluate the same 50 cases using identical prompts and scoring rules. Each judge evaluates each case independently in a single request, without tool access or context from other cases. The judges identify the primary source of difficulty, assess whether the available evidence is sufficient, and specify reasonable recovery actions together with a preferred action.

We evaluate agreement with OMH along two dimensions. *Action acceptability* measures whether the judge considers OMH's selected action a reasonable next step: a case counts as agreement if that action is included in the judge's set of reasonable actions. *Preferred-action agreement* measures whether the judge would choose exactly the same action as OMH, which is a stricter criterion. Each metric is reported as the number of matching cases over the corresponding audit subset.

Table 10: External audit of 50 stratified review and routing decisions. *Accepted* denotes that OMH's action belongs to the judge's reasonable-action set; *Preferred* denotes exact agreement with its preferred action.

		Claude Opus 4.8		**GPT-5.6 Terra**	
OMH action	**Decisions**	***Accepted***	***Preferred***	***Accepted***	***Preferred***
Continue formal proving	10	10	10	8	7
Revise informal guidance	7	2	0	4	3
Escalate to node review	8	8	8	7	7
Direct retry	7	7	7	7	5
Further decomposition	10	8	5	8	7
Parent-decomposition revision	8	7	7	7	7
Total	50	42	37	41	36

Across 50 decision points, two model families independently show substantial agreement with OMH's decisions. Claude Opus 4.8 and GPT-5.6 Terra consider the selected action reasonable in 42/50 and 41/50 cases, respectively, with both accepting 37/50 decisions. Their preferred actions exactly match OMH's choices in 37/50 and 36/50 cases (Table 10), which indicates that OMH's recovery decisions are consistent across different model families given the proof state, failure history, and surrounding structural context available at the time of review. Notably, neither judge marked

all three available actions as reasonable in any audited case, indicating that the accepted-action sets remained selective.

For local review, the external judges show strong agreement with OMH on most decisions. Claude Opus 4.8 and GPT-5.6 Terra accept 10/10 and 8/10 decisions to continue formal proving, respectively, and 8/8 and 7/8 decisions to escalate to node review. The main disagreement is concentrated on guidance revision, which they accept in 2/7 and 4/7 cases. Their rationales suggest that these cases often lie near the boundary between formal and strategic difficulty, where repeated formal failures or missing library support may justify either further refinement of the current approach or a change of proof strategy. One possible source of this variation is the diversity of viable proof strategies, which can lead different reviewers to favor different recovery choices.

Overall, the audit provides cross-model evidence that OMH's review decisions are consistent with the information available at the time of recovery. The observed agreement is also consistent with the intended role of the multi-level review mechanism in distinguishing recovery across formal proving, proof strategy, and proof structure.

Following are the prompts used for external audit.

Local-review audit prompt.

```
    You are auditing a LOCAL review decision in a Lean
theorem-proving process.

    The current lemma statement is fixed at this stage. The
possible next scopes
    of recovery are:

    - continue_formal_proving:
    Continue working on the current formal proof trajectory while
keeping the
    current informal guidance and statement.

    - revise_informal_guidance:
    Revise the informal mathematical guidance or proof strategy
while keeping
    the current formal statement fixed, then restart or redirect
formal proving.

    - escalate_to_node_review:
    Escalate the case for broader reconsideration beyond the
current local
    proving process.

    For the primary source of difficulty, use exactly one of:

    - formal_proving
    The statement and mathematical route remain broadly
appropriate, and the
    main difficulty lies in Lean realization, theorem/library
discovery,
    proof construction, compiler errors, or incomplete formal
details.

    - informal_guidance_error
    The current informal argument contains a substantive
mathematical mistake,
    missing step, or invalid reasoning that should be repaired.

    - strategy_route_unsuitable
    The statement may remain appropriate, but the current
mathematical route
```

```
    is poorly suited to formalization, unnecessarily difficult,
or an
    alternative proof strategy appears substantially more
appropriate.

    - statement_or_structure_issue
    The evidence suggests that the current statement is false,
unnecessarily
    strong, poorly scoped, or that the surrounding proof
structure may require
    broader reconsideration.

    - unclear
    The available evidence is insufficient to distinguish these
cases
    reliably.

    CASE:

    {{CASE_CONTEXT}}

    Return exactly one JSON object with this schema:

    {
            "primary_difficulty": "formal_proving |
informal_guidance_error | strategy_route_unsuitable |
statement_or_structure_issue | unclear",
            "evidence_sufficiency": "sufficient | insufficient",
            "reasonable_actions": [
            "continue_formal_proving | revise_informal_guidance |
escalate_to_node_review"
            ],
            "preferred_action": "continue_formal_proving |
revise_informal_guidance | escalate_to_node_review |
no_clear_preference",
            "rationale": "at most 80 words"
    }

    Rules:
    - reasonable_actions may contain one or more actions.
    - Do not include an action merely because it is possible;
include it only if
    it is defensible given the evidence.
    - If evidence_sufficiency is "insufficient",
reasonable_actions may be empty
    or may contain actions that remain defensible despite the
uncertainty.
    - preferred_action is the action you would choose if required
to select one.
    - If multiple actions are comparably justified, use
"no_clear_preference".
```

Node-level routing audit prompt.

```
    You are auditing a NODE-LEVEL routing decision in a Lean
theorem-proving
    process after local recovery has been insufficient.

    The possible next scopes of recovery are:
```

```
    - direct_retry:
    Keep the current statement unchanged and make another fresh
proving attempt,
    using the broader proof context and accumulated failure
history.

    - further_decomposition:
    Keep the current statement as the objective but introduce
simpler child
    lemmas whose verified proofs can establish it.

    - parent_decomposition_revision:
    Reconsider the decomposition that introduced the current node
because the
    current node may be unsuitable for its assigned role in the
surrounding
    proof structure.

    CASE:

    {{CASE_CONTEXT}}

    Return exactly one JSON object with this schema:

    {
            "primary_issue": "same_statement_retry_viable |
current_statement_benefits_from_decomposition |
parent_structure_problematic | unclear",
            "evidence_sufficiency": "sufficient | insufficient",
            "reasonable_actions": [
            "direct_retry | further_decomposition |
parent_decomposition_revision"
            ],
            "preferred_action": "direct_retry |
further_decomposition | parent_decomposition_revision |
no_clear_preference",
            "rationale": "at most 80 words"
    }

    Interpret the issue labels as follows:

    - same_statement_retry_viable:
    The current statement and surrounding decomposition remain
appropriate, and
    another direct attempt is defensible.

    - current_statement_benefits_from_decomposition:
    The statement remains appropriate, but proving it through
simpler
    intermediate lemmas appears more suitable than another direct
attempt.

    - parent_structure_problematic:
    The evidence suggests that the current node is unsuitable for
the role it
    plays in the parent decomposition, or that the surrounding
decomposition
    should be reconsidered.

    - unclear:
    The evidence is insufficient for a reliable distinction.

    Rules:
```

```
        - reasonable_actions may contain one or more actions.
        - The task is to identify reasonable actions, not a uniquely
    optimal one.
        - preferred_action is the action you would choose if required
    to select one.
        - If multiple actions are comparably justified, use
    "no_clear_preference".
```