HUMANIZE OLYMPIAD AGENTS: A MULTI-SESSION WORKFLOW FOR OLYMPIAD PROBLEM SOLVING WITH SOLVE, REVIEW, WIKI, VERIFY

Anonymous authors
Paper under double-blind review

ABSTRACT

Large language model (LLM) agents are rapidly improving on long-horizon scientific and olympiad tasks. Recent systems have produced a Lean-formalized resolution of the Navier–Stokes problem and autonomous solutions to four open questions in the Erdős Conjectures database (OpenAI, 2026b; Feng et al., 2026). Yet neither system releases both its underlying model and complete inference stack. More broadly, a single agent session remains brittle on difficult problems: an early error can propagate, missing domain knowledge can stall progress, and the agent can appear to succeed after misreading or weakening the task. We introduce **HOA (Humanize Olympiads Agent)**, an open, flow-level framework that coordinates multiple agent sessions through three complementary feedback channels. First, an independent reviewer checks each candidate and returns actionable feedback. Second, a structured wiki supplies task-specific knowledge that may be missing or unreliable in model pretraining. Third, a verifier checks generated Lean proofs against trusted challenge statements. With GPT-5.6 Sol, HOA reaches the **perfect score on IMO, IPhO, and IOI 2026**, and **near-full score on IChO 2026** problem set and solves **all 672 problems (100%) on PutnamBench**. HOA solves **216 of 302 problems (71.5%) on Lean-Eval**. Furthermore, we extend HOA's results with the open models – With Kimi-K3, HOA also achieves full score on IMO, IPhO,on IOI, and gold-level 97.1% on IChO, which closely matches the closed-source GPT-5.6 Sol + `/Goal` baseline on IMO and IPhO and exceeds it on IChO and IOI, showing that systematic feedback can compensate for substantial differences in base-model capability.

1 INTRODUCTION

LLM-based agents can now carry out long sequences of reasoning and action. They browse and synthesize evidence, develop and test code, construct mathematical arguments, and interact with formal proof assistants. Systems such as Deep Research, the AI Scientist, and AI co-scientist show how tools and iterative orchestration can turn a language model into a research workflow rather than a single-response generator (OpenAI, 2025; Lu et al., 2024; Gottweis et al., 2026). Their performance depends on both the base model and the *harness*: the prompts, tools, context management, execution environment, and control flow that allocate the inference budget, but all these works do not reveal those details. Community thereby needs open harnesses for those reproducible results.

Yet most evaluations treat a single agent session as the basic unit of problem solving. This design can be easily integrated into the next generation model and is fragile for hard research-level problems. First, a plausible but false intermediate claim can contaminate the rest of a long trajectory. Second, more reasoning cannot supply a theorem, scientific rule, or algorithmic pattern that the model does not know. Third, an agent may misread a requirement or weaken a goal—for example, by proving a nearby statement or testing convenient cases—and appear to succeed without solving the original task. We refer to this last failure mode as problem misunderstanding and reward hacking. These failures make longer single trajectories yield sharply diminishing returns.

We distinguish three levels of agents. At the *model level*, a direct API call returns an answer without an execution environment. At the *tool level*, a CLI-harness such as Codex and Claude Code gives the

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.

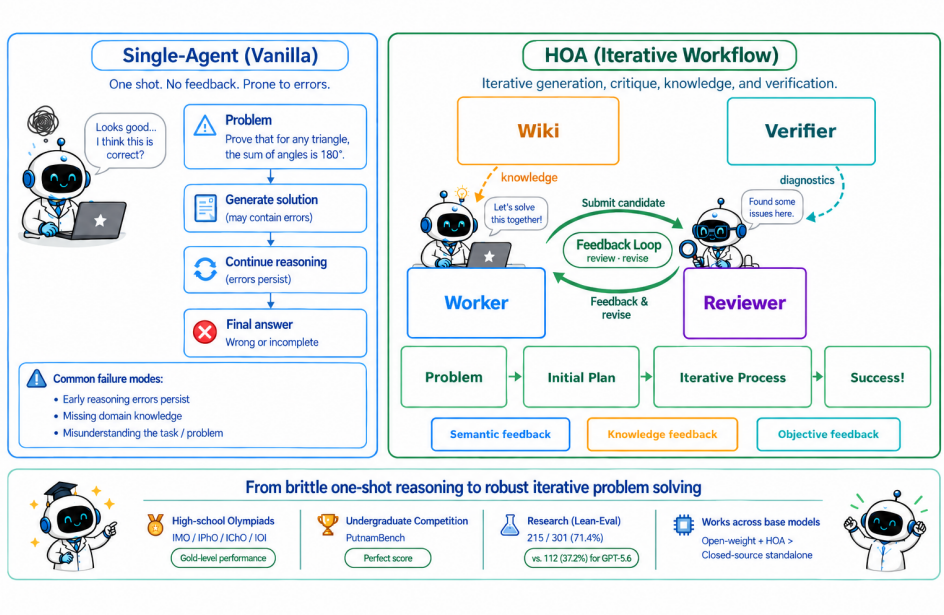


Figure 1: HOA replaces a brittle one-shot workflow with iterative review, knowledge retrieval, and verification. The bottom row summarizes results from high-school olympiads to research-level formal mathematics.

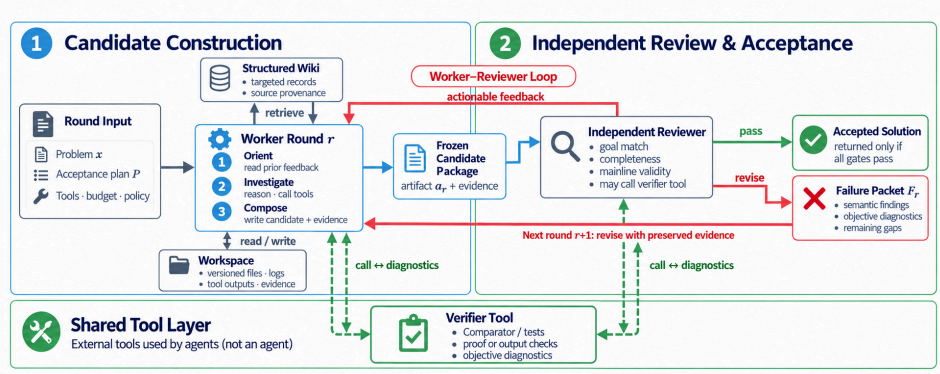


Figure 2: Overview of HOA. The worker receives the problem, acceptance plan, tools, budget, and prior feedback, and may retrieve provenance-bearing wiki records while building a versioned candidate. An independent reviewer inspects the frozen candidate and returns semantic findings. The worker and reviewer may both call the external verifier for objective checks. If any requirement fails, the findings and diagnostics start the next bounded worker round.

model tools to read files, execute code, and check intermediate results. At the *flow level*, a workflow coordinates multiple agent sessions and tools. Humanize is a flow-level protocol that extends Codex with an independent reviewer and a worker–reviewer revision loop (Liu et al., 2026b). The reviewer helps detect hallucinations, problem misunderstanding, and reward hacking.

We propose **HOA (Humanize Olympiads Agent)**, whose new contribution is an artifact-centered acceptance protocol that routes structured knowledge retrieval, independent semantic review, and task-specific objective verification through a shared worker–reviewer repair loop and accepts a candidate only after all applicable gates pass. A worker constructs a candidate artifact and consults a structured wiki when it identifies a knowledge gap. An independent reviewer checks the frozen artifact against the original problem and acceptance plan, then returns specific findings to the worker. Semantic review and objective validation remain separate gates. For mathematical tasks, the reviewer runs verifier on the unchanged candidate before accepting it (Lean FRO, 2026). A result is accepted when every required gate passes. Figure 1 and Figure. 2 explains how HOA solves a complex olympiad question.

The three feedback channels address different failure modes. Review catches omitted cases, invalid inferences, and solutions to the wrong problem. The wiki supplies missing knowledge. Verification provides an objective, task-specific signal. HOA keeps these roles separate while applying all feedback to the same evolving artifact. We evaluate HOA on high-school olympiads, college-level competition mathematics, and research-level formal mathematics, and HOA achieves top performance in various competitions and benchmarks.

Our contributions are:

- We introduce a flow-level worker–review framework for olympiads and frontier problems solving. We add task-specific verification tools, together with a structured wiki that supplies targeted, provenance-tracked knowledge, and thoroughly evaluate across various domains.
- We introduce a flow-level formalization agents that can turn physics problems and figures into typed Lean statements, checks them against the original problems, to enable later agent-driven proof.
- HOA reaches state-of-the-art results: **full scores on IMO, IPhO, IOI 2026**, and **near pefect score on IChO 2026**, all based on the latest competitions and there is no leakage. HOA also solved **672/672 on PutnamBench** including all problems from 1963 to 2023, and achiving the **#2 ranking on Lean-Eval**.
- HOA also shows the potential of open harness plus open models. We extensively test open source models and shows that HOA can boost the performance of open models and enables open-weight models to match or surpass standalone closed-source agents.

2 Related Work

2.1 Language Models for Formal Theorem Proving

LLM-based theorem proving in Lean (Moura & Ullrich, 2021) includes specialized provers, tool-using agents, and multi-stage systems. Specialized provers include TheoremLLaMA, Hunyuan-Prover, the DeepSeek-Prover series, the Goedel-Prover series, and Kimina-Prover (Wang et al., 2024; Li et al., 2025; Xin et al., 2024a;b; Ren et al., 2025; Lin et al., 2025b;c; Wang et al., 2025). Lean-STaR interleaves reasoning and proof tokens during generation (Lin et al., 2025a), while AlphaProof uses RL-based proof search at olympiad scale (Google DeepMind, 2024).

Tool-using agents query Lean during inference and revise failed proofs. AxProverBase and Numina-Lean-Agent use compact agent loops (Requena et al., 2026; Liu et al., 2026a), while LongCat-Flash-Prover integrates tool use into reinforcement learning (Wang et al., 2026). OProver repairs failed proofs with compiler feedback and retrieved examples (Ma et al., 2026), and VERITAS routes verifier feedback into critic-guided search (Acharya et al., 2026). Other systems add explicit proof plans: Draft-Sketch-Prove translates informal proofs into formal sketches (Jiang et al., 2023); Hilbert recursively decomposes goals (Varambally et al., 2026); Seed-Prover refines proof sketches (Chen et al., 2025b;a); Goedel-Architect revises a global dependency graph (**?**); and LEAP combines informal blueprints, decomposition, self-refinement, and Lean feedback (Kung et al., 2026). HOA adds an independent worker–reviewer loop, reusable knowledge, and task-specific verification across formal mathematics and multidisciplinary olympiad tasks.

2.2 Long-Horizon Research and Olympiad Agents

Research agents extend language models with browsing, code execution, and iterative control. Deep Research supports extended browsing and synthesis (OpenAI, 2025); the AI Scientist covers ideation, experimentation, writing, and simulated review (Lu et al., 2024); and AI co-scientist uses specialized generation, reflection, and ranking stages to refine hypotheses (Gottweis et al., 2026). AlphaEvolve pairs LLM-generated programs with evolutionary search and automatic evaluators (Novikov et al., 2025). MLE-bench further shows that both the base model and agent scaffold shape long-horizon engineering performance (Chan et al., 2025).

Olympiad and research agents increasingly combine generation with iterative verification. Aletheia uses a generator–verifier–reviser loop, browsing, and inference-time scaling (Feng et al., 2026). A model-agnostic verification-and-refinement pipeline reaches IMO gold-level performance (Huang

& Yang, 2025), while domain-specific scaffolds support chemistry and broader science olympiads (Xu et al., 2025). RMA combines literature search, a shared knowledge bank, and multi-round proposer–verifier feedback for research mathematics (Zhao et al., 2026). In programming, Gemini 2.5 Deep Think solved 10 of 12 problems at the 2025 ICPC World Finals, a gold-medal-level result that would have ranked second among the competing teams (Lin & Cheng, 2025). Generation, behavioral clustering, and ranking have also reached gold-level IOI 2025 performance with open-weight models (Samadi et al., 2026). HOA combines an independent worker–reviewer repair loop, a structured wiki, and task-specific verification in one workflow, then applies it across mathematics, physics, chemistry, and programming.

2.3 Prover Driven by Agent Loops

Formal provers increasingly use long-running agent loops. AxiomProver generated Lean proofs for all 12 Putnam 2025 problems, although its public release provides proof artifacts rather than a full system description (Axiom, 2026). Aleph Prover plans subproblems, checks them in Lean, and updates a global plan; paired with GPT-5.2, it reports solving 668 of 672 PutnamBench problems (Isenbaev & Hanin, 2026). These systems show that repeated planning and verification can scale formal reasoning.

HOA builds on Humanize (Liu et al., 2026b), in which an independent worker and reviewer repeat until explicit acceptance criteria are met. Related methods use self-critique, stored feedback, or tool-grounded critique (Madaan et al., 2023; Shinn et al., 2023; Gou et al., 2024), but self-correction can fail without a reliable external signal (Huang et al., 2024). HOA therefore separates generation, review, and verification: the reviewer provides evidence and specific revision requests, and the worker resubmits the revised artifact. For Lean tasks, Comparator builds the submission in a sandbox, matches it against the trusted statement, restricts permitted axioms, and checks the proof with the Lean kernel (Lean FRO, 2026). These checks matter because a proof can compile while using a disallowed axiom (Vamshi & Yang, 2026).

Agent loops also benefit from external knowledge. Retrieval-augmented generation, ReAct, and RARR combine model reasoning with retrieved evidence and external actions (Lewis et al., 2021; Yao et al., 2023; Gao et al., 2023); LeanDojo and ReProver retrieve premises from Lean libraries (Yang et al., 2023). HOA instead uses a bounded, task-specific wiki. Following Tau Ceti, Lean Pool, and DreamProver (Tau Ceti Project, 2026; Ilin & Asher, 2026; Zhang et al., 2026), its mathematics wiki turns accepted proofs into provenance-tracked lemma records. Its chemistry wiki draws from textbooks, olympiad problems outside the evaluation set, and online references. Each run retrieves a small set of relevant records.

3 Method

3.1 Overview

HOA is a flow-level agent framework built on Humanize (Liu et al., 2026b). It contains a *worker*, an independent *reviewer*, a structured *wiki*, and a task-specific *verifier*. Let x denote the original problem and a_t the artifact produced at iteration t. The worker constructs a_t from x and prior feedback, consulting the wiki when it can state a specific knowledge need. The reviewer checks semantic correctness, while the verifier checks a machine-evaluable condition. For mathematics and physics tasks, HOA uses Comparator as the verifier (Lean FRO, 2026). A candidate counts as solved after every required gate accepts it.

HOA combines three forms of feedback. Semantic feedback asks whether the artifact fully answers the original problem. Knowledge feedback supplies missing facts or methods. Objective feedback tests a machine-evaluable condition. These signals are complementary rather than interchangeable.

3.2 Worker–Reviewer Loop

The worker–reviewer loop separates artifact construction from acceptance. In each round, the worker produces or revises a candidate. An independent reviewer then evaluates the frozen candidate against the original problem and acceptance criteria. Unresolved findings return to the worker

as actionable feedback for the next round. The loop ends when the reviewer declares the candidate complete or the run reaches its resource limits. This separation lets the worker explore and repair without judging its own output.

The worker maintains the evolving artifact. In the first round, it receives the problem, task instructions, available tools, and acceptance criteria. In later rounds, it also receives the current artifact, reviewer findings, and verifier diagnostics. The worker must address every reported issue. When it identifies a knowledge gap, it sends a focused wiki query, incorporates the retrieved material with provenance, and revises the artifact. It may call the verifier for immediate machine feedback, but it cannot accept its own output or bypass review.

The reviewer receives the original problem, candidate artifact, acceptance plan, and a runner-generated summary of the current round. It does not see the worker's hidden reasoning or conversation history, and the candidate remains read-only during review. This separation reduces anchoring and prevents silent repairs. The reviewer instead returns findings that guide the next worker round.

3.3 Wikis for Domain-Specific Knowledge

HOA uses separate wikis for mathematics and chemistry because the two domains require different forms of reusable knowledge.

Mathematics wiki. The mathematics wiki follows the reusable formal libraries developed by Tau Ceti and Lean Pool (Tau Ceti Project, 2026; Ilin & Asher, 2026). After a mathematical problem is solved and accepted, a curation agent extracts the main theorem and reusable intermediate lemmas from the Lean development. Each record contains declaration names, Lean signatures, a short description, the source problem, required imports, source locations, and verification status.

Chemistry wiki. The chemistry wiki is built before evaluation from textbooks, olympiad problems outside the test set, and relevant online references. Agents organize the material into concept-centered pages that cover definitions, reactions, formulas, experimental procedures, problem types, and solution patterns. Each page records its sources.

We do not build separate wikis for physics or coding. IPhO and IOI required knowledges are largely already presented in model pretraining. We focus the wiki on mathematics and chemistry, where structured external knowledge provides greater value.

When the worker identifies a missing fact or method, it sends a focused query. The retrieval layer searches Lean declarations in the mathematics wiki or concept pages in the chemistry wiki, then returns a small set of records with source identifiers.

3.4 Physics Problem Formalization

We formalize each physics problem as a Lean theorem before asking the worker to prove it. We use the open-source Science Mango pipeline (Humanfia, 2026), which receives the problem text and figures. The pipeline first creates a blueprint that separates the physical laws, given values, results from earlier parts, and the claim to prove.

An autoformalization agent grounds this blueprint in existing formal mathematics by querying LeanExplore (Asher, 2025), a search engine for Lean 4 declarations, over Mathlib (mathlib Community, 2020) and PhysLean (Tooby-Smith, 2024). It then writes typed Lean declarations with sorry proof bodies. It preserves the roles and dimensions of physical quantities. When no library definition exists, the agent introduces a small local abstraction that retains the physical meaning. The target equation is withheld before being presented to agents for solving. If the model cannot write the physical laws as equations, it first produces a natural-language derivation and then autoformalizes that derivation. A physics reviewer compares the Lean statement with the original problem and rejects missing laws, omitted figure parameters, tautologies, and statements that assume the answer. Once the statement compiles and passes review, it is frozen. The prover may fill the proof body but cannot weaken the statement; semantic errors return for redrafting.

3.5 VERIFIER

The verifier determines whether the current candidate satisfies a machine-evaluable acceptance condition. Its exit status and diagnostic log are retained as runner evidence. When verification fails, the calling agent receives the diagnostic summary, and the candidate cannot be accepted.

For mathematical tasks and formalized physics task, the verifier is Comparator (Lean FRO, 2026). , which checks untrusted Lean submissions against trusted challenge statements. Comparator builds the candidate in a sandbox, confirms that the submitted theorem matches the trusted challenge, restricts permitted axioms, and validates the exported proof with the Lean kernel. Every mathematical problem is encoded as a Lean theorem and counts as solved when Comparator accepts the candidate proof. The reviewer remains responsible for detecting reward hacking and problem misunderstanding; the verifier checks whether the stated theorem is proven. For coding task, the worker and reviewer can create test cases and execute the code to see if the time limit is satisfied and if the answer is correct.

The reviewer and verifier play different roles. The reviewer is an agent in the worker–reviewer loop: it assesses the artifact semantically, identifies reward hacking or problem misunderstanding, and decides whether another revision is needed. The verifier is an objective, task-specific tool rather than an agent in the loop. Both the worker and reviewer can call it when they need machine-evaluable evidence about the current candidate.

4 EXPERIMENTS

4.1 EXPERIMENTAL SETUP

We evaluate HOA across tasks that produce different artifacts and use different verification signals.

Benchmarks. Our mathematics evaluation spans three levels. At the high-school olympiad level, we use Axiom Math's formalized versions of the six IMO 2026 problems (Axiom Math, 2026) with a stricter binary scoring rule. A solution receives full credit if it is mathematically correct and formalized as a Lean proof accepted by Comparator and the Lean kernel; otherwise, it receives zero points. We compare the total with the gold-medal cutoff. At the undergraduate level, we use the frozen 672-problem Lean subset of PutnamBench (Tsoukalas et al., 2024). At the research level, we use the 302 formal theorem-proving tasks in Lean-Eval. PutnamBench and Lean-Eval use the same Comparator and Lean-kernel acceptance rule.

We also evaluate physics, chemistry, and competitive programming on the official 2026 IPhO, IChO, and IOI problems. IPhO and IChO responses are graded by an LLM judge according to their official scoring rubrics, and human reviewers check the resulting grades. IOI submissions are graded by Codeforces against hidden test cases. We compare each competition score with its 2026 gold-medal cutoff. The ablation study additionally uses a 50-problem Physics Cup subset.

Models and baselines. Our main closed-source model is GPT-5.6. We compare a standalone model (pure api call), a standalone agent (with codex harness), a standalone agent with wiki and verifier, and full HOA. Cross-model experiments use the open-weight Hugging Face checkpoints `moonshotai/Kimi-K3`, `zai-org/GLM-5.3`, and `deepseek-ai/DeepSeek-V4-Pro` (Moonshot AI, 2026; Z.ai, 2026; DeepSeek-AI, 2026). We also compare with public Seed-Prover and AxiomProver results when the task set and acceptance rule match.

Metrics. For binary tasks, we report solved count and pass rate. For scored competitions, we report official raw points and whether the score reaches that year's gold threshold. When the sample size permits, we report bootstrap confidence intervals over problems. Failures, timeouts, and budget exhaustion remain in the denominator.

4.2 MAIN RESULTS

Table 1 shows consistent gains across all three evaluation levels. **HOA meets or exceeds the 2026 gold thresholds on IMO, IPhO, IChO, and IOI** and solves **672/672 PutnamBench problems**. On Lean-Eval, HOA solves **216/302 problems (71.5%)**, compared with **112/302 (37.1%)** for the

Table 1: HOA results across three evaluation levels. HOA reaches the 2026 gold threshold on IMO, IPhO, IChO, and IOI; solves all 672 PutnamBench problems (Tsoukalas et al., 2024); and improves Lean-Eval from GPT-5.6's official 112/302 to 216/302, ranking #2 (Lean Prover Community, 2026). The Lean kernel verifies every mathematics result. ✓ marks an open-source release, and ✗ marks a closed-source system.

Benchmark	System	Open Source	Solved / Total
High-School-Level Competitions—Commercial Models GPT-5.6-Sol			
IMO 2026	HOA (ours)	✓	6 / 6
IPhO 2026	HOA (ours)	✓	30 / 30
IChO 2026	HOA (ours)	✓	97.2 / 100
IOI 2026	HOA (ours)	✓	600 / 600
High-School-Level Competitions—Open Models Kimi-K3			
IMO 2026	HOA (ours)	✓	6 / 6
IPhO 2026	HOA (ours)	✓	30 / 30
IChO 2026	HOA (ours)	✓	97.1 / 100
IOI 2026	HOA (ours)	✓	600 / 600
College Level Competitions			
PutnamBench	HOA (ours)	✓	**672 / 672**
PutnamBench	Harmonic (Achim et al., 2025)	✗	**672 / 672**
PutnamBench	Aleph Prover (Isenbaev & Hanin, 2026)	✗	668 / 672
PutnamBench	Mistral Prover (Mistral AI, 2026)	✓	658 / 672
PutnamBench	Goedel-Architect (Chung et al., 2026)	✓	642 / 672
PutnamBench	Seed-Prover 1.5 (Chen et al., 2025a)	✗	581 / 672
Research Level Problems			
Lean-Eval	AxiomProver (Axiom, 2026)	✗	**218 / 302**
Lean-Eval	HOA (ours)	✓	216 / 302
Lean-Eval	Tau (caj.al) (Cajal Technologies, 2026)	✗	200 / 302
Lean-Eval	Aristotle (Harmonic) (Achim et al., 2025)	✗	162 / 302
Lean-Eval	Seed-Prover (Chen et al., 2025b)	✗	156 / 302
Lean-Eval	GPT-5.6 official (OpenAI, 2026a)	✗	112 / 302

official GPT-5.6 result—a gain of **104 proofs and 34.4 percentage points**. Every formal proof is checked by the Lean kernel. Although the scoring rules differ across fields, the same pattern holds: review and revision convert failed attempts into accepted solutions.

4.3 An Open Model with an Open Harness Can Match Closed-Source Agents

Table 2 shows that HOA consistently improves performance across base models. At the pure Codex level, the open models trail the closed GPT-5.6 model on most competitions; with HOA, Kimi K3 achieves full marks on IOI, IMO, and IPhO and nearly matches GPT-5.6 on IChO. Figure 3 shows the same convergence on PutnamBench. Overall, the gap between open-model + HOA and closed-model + HOA is substantially narrower than the gap between their pure-Codex counterparts.

4.4 Ablation Study

We compare three levels of scaffolding—a direct API call, the original Codex harness, and the full HOA workflow—with GPT-5.6, Kimi K3, GLM-5.3, and DeepSeek V4 Pro. Table 3 groups the results by base model and reports solved counts on PutnamBench and Physics Cup. HOA gives the best result for every model on both evaluations. The PutnamBench results for these configurations remain provisional.

Table 2: Cross-model results for HOA and pure Codex on the 2026 IOI, IMO, IPhO, and IChO benchmarks. HOA improves performance across base models, with Kimi K3 + HOA nearly matching GPT-5.6 + HOA. Colored subscripts show absolute changes from pure Codex.

Model	IOI	IMO	IPhO	IChO
GPT-5.6-sol (size undisclosed)				
+ /Goal	562/600	**6/6**	**30/30**	87.8
+ HOA	**600/600**$_{+38}$	**6/6**$_{0}$	**30/30**$_{0}$	**97.2**$_{+9.4}$
Kimi K3 (2.8T)				
+ /Goal	400/600	4/6	**30/30**	65.0
+ HOA	**600/600**$_{+200}$	**6/6**$_{+2}$	**30/30**$_{0}$	97.1$_{+32.1}$
DeepSeek V4 Pro (1.6T)				
+ /Goal	347/600	4/6	28.4/30	58.7
+ HOA	411/600$_{+64}$	4/6$_{0}$	28.8/30$_{+0.4}$	60.7$_{+2.0}$
GLM-5.3 (744B)				
+ /Goal	249/600	2/6	28.3/30	54.8
+ HOA	428/600$_{+179}$	4/6$_{+2}$	29.9/30$_{+1.6}$	58.8$_{+4.8}$

Table 3: Model-, tool-, and flow-level results on 50-problem subsets of PutnamBench and Physics Cup. HOA performs best for every base model on both evaluations; colored subscripts show gains over the model-level API baseline.

Model / Level	PutnamBench	Physics Cup
GPT-5.6-sol		
Model-Level (API)	3	31
Tool-Level (CLI)	46$_{+43}$	41$_{+10}$
Flow-Level (HOA)	**50**$_{+47}$	**44**$_{+13}$
Kimi-K3		
Model-Level (API)	1	35
Tool-Level (CLI)	4$_{+3}$	37$_{+2}$
Flow-Level (HOA)	**47**$_{+46}$	**42**$_{+7}$
GLM-5.3		
Model-Level (API)	0	24
Tool-Level (CLI)	2$_{+2}$	34$_{+10}$
Flow-Level (HOA)	**25**$_{+25}$	**40**$_{+16}$
DeepSeek V4 Pro		
Model-Level (API)	0	32
Tool-Level (CLI)	4$_{+4}$	35$_{+3}$
Flow-Level (HOA)	**13**$_{+13}$	**39**$_{+7}$

4.5 Analysis

Figure 4 compares cumulative Lean-Eval accuracy against worker time for HOA and Pure Codex with `/Goal`. The evaluation uses the first 100 eligible problems in alphabetical order and retains every failure. Pure Codex + `/Goal` reaches 34/100, while HOA continues recovering verified solutions and reaches 63/100. We launch HOA and Codex for 150 hours in lean eval experiment. Pure Codex with `/Goal` gets saturated and makes no progress after about 30 hours, while HOA and continuouly make progress throughout the process. The worker–reviewer loop turns this additional time into verified solutions instead of letting the worker try the same method and always make the same mistake.

Wiki reuse in a Lean trajectory. The G_2 tensor-square decomposition run shows how the mathematics wiki acts as an executable interface rather than an unstructured hint. During the trajectory, the agent retrieved three records that had passed Comparator and independent review. They formal-

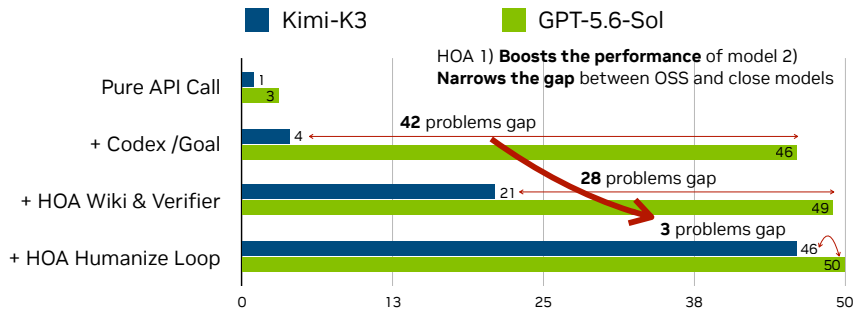
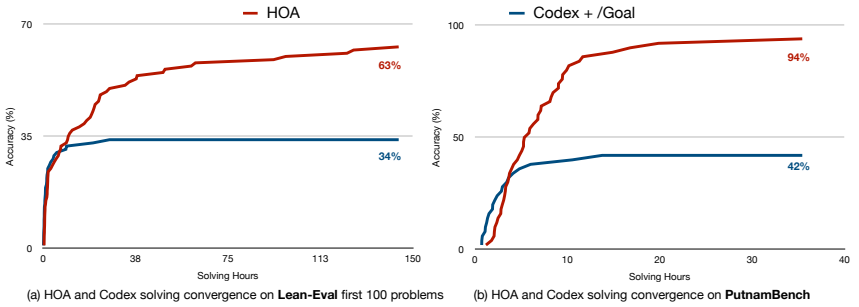


Figure 3: PutnamBench accuracy across four system levels for Kimi K3 and GPT-5.6 . PutnamBench contains more chanllenging problems than IMO. HOA boosts Kimi K3 from 1/50 to 46/50 and GPT-5.6 from 3/50 to 50/50. HOA greatly narrows the gap between open source models and close ones (from 42 problesm gap to 3 problems gap).



(a) HOA and Codex solving convergence on **Lean-Eval** first 100 problems (b) HOA and Codex solving convergence on **PutnamBench**

Figure 4: Cumulative Lean-Eval accuracy versus worker time on the same 100-problem subset. Pure Codex + `/Goal` plateaus at 34/100 after about 30 hours, while HOA reaches 63/100 over 150 hours. HOA provides another scaling dimension: sustained agent time. Its review and revision loop converts longer runs into verified solutions by recovering from intermediate failures. All failures remain in the denominator. HOA's full Lean-Eval result is 216/302 (71.5%), ranking #2.

ize a 14-dimensional simple G_2 quotient, a 64-dimensional irreducible highest-weight module, and the classification of fourteen irreducible tensor-square types. The agent applied these Lean declarations directly when assembling the final theorem instead of reproducing their proofs. The resulting artifact passed Comparator and both the Nanoda and default Lean kernels. This example shows how verified lemma records turn completed subproblems into typed, provenance-tracked building blocks for later formalization steps.

5 Conclusion

Reliable long-horizon problem solving requires more than a promising initial idea. An agent must detect errors, acquire missing knowledge, preserve the original specification, and determine when the task is complete. HOA addresses these needs with a mandatory worker–reviewer repair loop, targeted wiki retrieval, and task-specific verification. Comparator checks mathematical candidates against trusted Lean statements, while programs retain their native execution tests.

HOA reaches gold-medal-level performance on the 2026 IMO, IPhO, IChO, and IOI problems and receives full credit on PutnamBench. On research-level mathematics, it improves GPT-5.6 from 112/302 to 216/302 solved Lean-Eval tasks, ranking #2. An open-weight model with HOA also outperforms a closed-source standalone agent. These results show that workflow design is a major source of capability rather than a thin layer around the base model.

Future work will explore parallel workflows and divide-and-conquer strategies that split complex problems into smaller parts, solve them independently, and combine their results.

AI USE STATEMENT

In this work, we used generative AI tools only for language polishing, grammar checking, and generating portions of the code. The research ideas, methodology, experimental design, and interpretation of results were developed by the authors. We reviewed all AI-assisted text to ensure that it accurately conveyed our intended meaning. LLM-generated code was verified and tested for correctness by authors. We take full responsibility for the final content of this work, including all text, claims, and artifacts produced with the assistance of generative AI.

ETHICS STATEMENT

The experiments use public benchmark and competition problems and involve no human subjects or personal data. We will audit evaluation items for possible training contamination and report the resulting uncertainty. Benchmark performance should not be interpreted as evidence of broad scientific autonomy

REPRODUCIBILITY STATEMENT

We provide an anonymous HOA code supplement containing the workflow implementation, native agent runtime, benchmark integration, the evaluation inventory, prompts, orchestration code, reviewer schema, wiki snapshot and source manifest, verifier specification, model and tool versions, budgets, per-run logs, and generated Lean proof artifacts. Section 4 lists the detail of every benchmark and model checkpoint. Now the script for our experiment is available at `https://anonymous.4open.science/r/hoa-submission-E2F8/README.md`

REFERENCES

Manish Acharya, Zhenyu Liao, Yueke Zhang, Kevin Leach, Yu Huang, and Yifan Zhang. Veritas: Verifier-guided proof search for zero-shot formal theorem proving, 2026. URL https://arxiv.org/abs/2606.19399.

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, 2025. URL https://arxiv.org/abs/2510.01346.

Justin Asher. Leanexplore: A search engine for lean 4 declarations. *arXiv preprint arXiv:2506.11085*, 2025.

Axiom. From seeing why to checking everything: AxiomProver solves all problems at Putnam 2025. https://axiommath.ai/research/from-seeing-why-to-checking-everything/, 2026.

Axiom Math. AxiomProver at IMO 2026. https://github.com/AxiomMath/IMO2026, 2026.

Cajal Technologies. Tau-prover v1. https://caj.al/blog/tau-prover-v1, 2026.

Jun Shern Chan, Neil Chowdhury, Oliver Jaffe, James Aung, Dane Sherburn, Evan Mays, Giulio Starace, Kevin Liu, Leon Maksin, Tejal Patwardhan, Lilian Weng, and Aleksander Madry. Mle-bench: Evaluating machine learning agents on machine learning engineering, 2025. URL https://arxiv.org/abs/2410.07095.

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, 2025a. URL https://arxiv.org/abs/2512.17260.

Luoxin Chen, Jinming Gu, Liankai Huang, Wenhao Huang, Zhicheng Jiang, Allan Jie, Xiaoran Jin, Xing Jin, Chenggang Li, Kaijing Ma, Cheng Ren, Jiawei Shen, Wenlei Shi, Tong Sun, He Sun, Jiahui Wang, Siran Wang, Zhihong Wang, Chenrui Wei, Shufa Wei, Yonghui Wu, Yuchen Wu, Yihang Xia, Huajian Xin, Fan Yang, Huaiyuan Ying, Hongyi Yuan, Zheng Yuan, Tianyang Zhan, Chi Zhang, Yue Zhang, Ge Zhang, Tianyun Zhao, Jianqiu Zhao, Yichi Zhou, and Thomas Hanwen Zhu. Seed-prover: Deep and broad reasoning for automated theorem proving, 2025b. URL https://arxiv.org/abs/2507.23726.

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, 2026. URL https://arxiv.org/abs/2606.06468.

DeepSeek-AI. DeepSeek-V4-Pro model card. https://huggingface.co/deepseek-ai/DeepSeek-V4-Pro, 2026.

Tony Feng, Trieu H. Trinh, Garrett Bingham, Dawsen Hwang, Yuri Chervonyi, Junehyuk Jung, Joonkyung Lee, Carlo Pagano, Sang hyun Kim, Federico Pasqualotto, Sergei Gukov, Jonathan N. Lee, Junsu Kim, Kaiying Hou, Golnaz Ghiasi, Yi Tay, YaGuang Li, Chenkai Kuang, Yuan Liu, Hanzhao Lin, Evan Zheran Liu, Nigamaa Nayakanti, Xiaomeng Yang, Heng-Tze Cheng, Demis Hassabis, Koray Kavukcuoglu, Quoc V. Le, and Thang Luong. Towards autonomous mathematics research, 2026. URL https://arxiv.org/abs/2602.10177.

Luyu Gao, Zhuyun Dai, Panupong Pasupat, Anthony Chen, Arun Tejasvi Chaganty, Yicheng Fan, Vincent Y. Zhao, Ni Lao, Hongrae Lee, Da-Cheng Juan, and Kelvin Guu. Rarr: Researching and revising what language models say, using language models, 2023. URL https://arxiv.org/abs/2210.08726.

Google DeepMind. AI achieves silver-medal standard solving international mathematical olympiad problems. https://deepmind.google/discover/blog/ai-solves-imo-problems-at-silver-medal-level/, 2024.

Juraj Gottweis, Wei-Hung Weng, Alexander Daryin, Tao Tu, Petar Sirkovic, Artiom Myaskovsky, Grzegorz Glowaty, Felix Weissenberger, Alessio Orlandi, Dan Popovici, Anil Palepu, Keran Rong, Ryutaro Tanno, Khaled Saab, Fan Zhang, Jacob Blum, Andrew Carroll, Kavita Kulkarni, Nenad Tomašev, Dina Zverinski, Ivor Rendulic, Elahe Vedadi, Florian Hasler, Luka Rimanic, Marina Boia, Ivan Budiselic, Ben Feinstein, Mathias Bellaiche, Tom Sheffer, Jan Freyberg, Jeremy Ratcliff, Ottavia Bertolli, Katherine Chou, Avinatan Hassidim, Burak Gokturk, Amin Vahdat, Yuan Guan, Vikram Dhillon, Eeshit Dhaval Vaishnav, Byron Lee, Tiago R. D. Costa, José R. Penadés, Gary Peltz, Yossi Matias, James Manyika, Demis Hassabis, Yunhan Xu, Pushmeet Kohli, Annalisa Pawlosky, Alan Karthikesalingam, and Vivek Natarajan. Accelerating scientific discovery with co-scientist. *Nature*, 655(8122):487–496, May 2026. ISSN 1476-4687. doi: 10.1038/s41586-026-10644-y. URL http://dx.doi.org/10.1038/s41586-026-10644-y.

Zhibin Gou, Zhihong Shao, Yeyun Gong, Yelong Shen, Yujiu Yang, Nan Duan, and Weizhu Chen. Critic: Large language models can self-correct with tool-interactive critiquing, 2024. URL https://arxiv.org/abs/2305.11738.

Jie Huang, Xinyun Chen, Swaroop Mishra, Huaixiu Steven Zheng, Adams Wei Yu, Xinying Song, and Denny Zhou. Large language models cannot self-correct reasoning yet, 2024. URL https://arxiv.org/abs/2310.01798.

Yichen Huang and Lin F. Yang. Winning gold at imo 2025 with a model-agnostic verification-and-refinement pipeline, 2025. URL https://arxiv.org/abs/2507.15855.

Humanfia. Science Mango: Physics problem formalization in Lean. https://github.com/humanfia/science-mango, 2026.

Vasily Ilin and Justin Asher. Lean Pool. https://github.com/Vilin97/lean-pool, 2026.

Vlad Isenbaev and Boris Hanin. Logical Intelligence's Aleph Solves PutnamBench. https://logicalintelligence.com/blog/aleph-solves-putnambench, 2026.

Albert Q. Jiang, Sean Welleck, Jin Peng Zhou, Wenda Li, Jiacheng Liu, Mateja Jamnik, Timothée Lacroix, Yuhuai Wu, and Guillaume Lample. Draft, sketch, and prove: Guiding formal theorem provers with informal proofs, 2023. URL https://arxiv.org/abs/2210.12283.

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, 2026. URL https://arxiv.org/abs/2606.03303.

Lean FRO. Comparator: A trustworthy judge for Lean proofs. https://github.com/leanprover/comparator, 2026.

Lean Prover Community. Lean-Eval. https://github.com/leanprover/lean-eval, 2026.

Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen tau Yih, Tim Rocktäschel, Sebastian Riedel, and Douwe Kiela. Retrieval-augmented generation for knowledge-intensive nlp tasks, 2021. URL https://arxiv.org/abs/2005.11401.

Yang Li, Dong Du, Linfeng Song, Chen Li, Weikang Wang, Tao Yang, and Haitao Mi. Hunyuanprover: A scalable data synthesis framework and guided tree search for automated theorem proving, 2025. URL https://arxiv.org/abs/2412.20735.

Hanzhao Lin and Heng-Tze Cheng. Gemini achieves gold-medal level at the 2025 ICPC world finals. Google DeepMind blog, 2025.

Haohan Lin, Zhiqing Sun, Sean Welleck, and Yiming Yang. Lean-star: Learning to interleave thinking and proving, 2025a. URL https://arxiv.org/abs/2407.10040.

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, 2025b. URL https://arxiv.org/abs/2502.07640.

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, 2025c. URL https://arxiv.org/abs/2508.03613.

Junqi Liu, Zihao Zhou, Zekai Zhu, Marco Dos Santos, Weikun He, Jiawei Liu, Ran Wang, 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, 2026a. URL https://arxiv.org/abs/2601.14027.

Sihao Liu, Ligeng Zhu, Zijian Zhang, Changye Li, Song Han, and Tony Nowatzki. Humanize: Orchestrate, execute, and observe agent flows. https://github.com/humanfia/humanize, 2026b. GitHub repository.

Chris Lu, Cong Lu, Robert Tjarko Lange, Jakob Foerster, Jeff Clune, and David Ha. The ai scientist: Towards fully automated open-ended scientific discovery, 2024. URL https://arxiv.org/abs/2408.06292.

David Ma, Kaijing Ma, Shawn Guo, Yunfeng Shi, Enduo Zhao, Jiajun Shi, Zhaoxiang Zhang, Gavin Cheung, Jiaheng Liu, and Zili Wang. Oprover: A unified framework for agentic formal theorem proving, 2026. URL https://arxiv.org/abs/2605.17283.

Aman Madaan, Niket Tandon, Prakhar Gupta, Skyler Hallinan, Luyu Gao, Sarah Wiegreffe, Uri Alon, Nouha Dziri, Shrimai Prabhumoye, Yiming Yang, Shashank Gupta, Bodhisattwa Prasad Majumder, Katherine Hermann, Sean Welleck, Amir Yazdanbakhsh, and Peter Clark. Self-refine: Iterative refinement with self-feedback, 2023. URL https://arxiv.org/abs/2303.17651.

The mathlib Community. The lean mathematical library. In *Proceedings of the 9th ACM SIGPLAN International Conference on Certified Programs and Proofs*, pp. 367–381. ACM, January 2020. doi: 10.1145/3372885.3373824. URL http://dx.doi.org/10.1145/3372885.3373824.

Mistral AI. Leanstral. https://www.alphaxiv.org/abs/2608.leanstralv1, 2026.

Moonshot AI. Kimi K3 model card. https://huggingface.co/moonshotai/Kimi-K3, 2026.

Leonardo de Moura and Sebastian Ullrich. The lean 4 theorem prover and programming language. In André Platzer and Geoff Sutcliffe (eds.), *Automated Deduction – CADE 28*, pp. 625–635, Cham, 2021. Springer International Publishing. ISBN 978-3-030-79876-5.

Alexander Novikov, Ngân Vũ, Marvin Eisenberger, Emilien Dupont, Po-Sen Huang, Adam Zsolt Wagner, Sergey Shirobokov, Borislav Kozlovskii, Francisco J. R. Ruiz, Abbas Mehrabian, M. Pawan Kumar, Abigail See, Swarat Chaudhuri, George Holland, Alex Davies, Sebastian Nowozin, Pushmeet Kohli, and Matej Balog. Alphaevolve: A coding agent for scientific and algorithmic discovery, 2025. URL https://arxiv.org/abs/2506.13131.

OpenAI. Introducing deep research. https://openai.com/index/introducing-deep-research/, 2025.

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

OpenAI. On the Navier–Stokes millennium prize problem. https://openai.com/index/navier-stokes-solution/, 2026b.

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, 2025. URL https://arxiv.org/abs/2504.21801.

Borja Requena, Austin Letson, Krystian Nowakowski, Izan Beltran-Ferreiro, and Leopoldo Sarra. A minimal agent for automated theorem proving, 2026. URL https://arxiv.org/abs/2602.24273.

Mehrzad Samadi, Aleksander Ficek, Sean Narenthiran, Siddhartha Jain, Wasi Uddin Ahmad, Somshubra Majumdar, Vahid Noroozi, and Boris Ginsburg. Scaling test-time compute to achieve ioi gold medal with open-weight models, 2026. URL https://arxiv.org/abs/2510.14232.

Noah Shinn, Federico Cassano, Edward Berman, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. Reflexion: Language agents with verbal reinforcement learning, 2023. URL https://arxiv.org/abs/2303.11366.

Tau Ceti Project. Tau Ceti: An agent-maintained library of formal mathematics. https://github.com/TauCetiProject/TauCeti, 2026.

Joseph Tooby-Smith. Heplean: Digitalising high energy physics, 2024. URL https://arxiv.org/abs/2405.08863.

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, 2024. URL https://arxiv.org/abs/2407.11214.

Bodla Krishna Vamshi and Haizhao Yang. Reward-oracle mcts for formal theorem proving: Sample-efficient search and the need for kernel-level proof auditing, 2026. URL https://arxiv.org/abs/2608.28639.

Sumanth Varambally, Thomas Voice, Yanchao Sun, Zhifeng Chen, Rose Yu, and Ke Ye. Hilbert: Recursively building formal proofs with informal reasoning, 2026. URL https://arxiv.org/abs/2509.22819.

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, 2025. URL https://arxiv.org/abs/2504.11354.

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, 2026. URL https://arxiv.org/abs/2603.21065.

Ruida Wang, Jipeng Zhang, Yizhen Jia, Rui Pan, Shizhe Diao, Renjie Pi, and Tong Zhang. Theoremllama: Transforming general-purpose llms into lean4 experts, 2024. URL https://arxiv.org/abs/2407.03203.

Huajian Xin, Daya Guo, Zhihong Shao, Zhizhou Ren, Qihao Zhu, Bo Liu, Chong Ruan, Wenda Li, and Xiaodan Liang. Deepseek-prover: Advancing theorem proving in llms through large-scale synthetic data, 2024a. URL https://arxiv.org/abs/2405.14333.

Huajian Xin, Z. Z. Ren, Junxiao Song, Zhihong Shao, Wanjia Zhao, Haocheng Wang, Bo Liu, Liyue Zhang, Xuan Lu, Qiushi Du, Wenjun Gao, 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, 2024b. URL https://arxiv.org/abs/2408.08152.

Qiang Xu, Shengyuan Bai, Leqing Chen, Zijing Liu, and Yu Li. Chemlabs on chemo: A multi-agent system for multimodal reasoning on icho 2025, 2025. URL https://arxiv.org/abs/2511.16205.

Kaiyu Yang, Aidan M. Swope, Alex Gu, Rahul Chalamala, Peiyang Song, Shixing Yu, Saad Godil, Ryan Prenger, and Anima Anandkumar. Leandojo: Theorem proving with retrieval-augmented language models, 2023. URL https://arxiv.org/abs/2306.15626.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. React: Synergizing reasoning and acting in language models, 2023. URL https://arxiv.org/abs/2210.03629.

Z.ai. GLM-5.3 model card. https://huggingface.co/zai-org/GLM-5.3, 2026.

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, 2026. URL https://arxiv.org/abs/2604.26311.

Zelin Zhao, Bo Yuan, Jaemoo Choi, and Yongxin Chen. Rma: an agentic system for research-level mathematical problems, 2026. URL https://arxiv.org/abs/2605.22875.

A Appendix

A.1 Evaluation Inventory

The camera-ready version will list every evaluated problem with its benchmark version, competition year, public release date, maximum score, HOA score, verifier type, Lean artifact identifier for mathematical problems, and contamination-audit status. This inventory makes claims such as "perfect score" and "gold-medal-level" precise.

A.2 Reviewer Record Template

Each reviewer record follows the schema below:

```
verdict: accept | revise
findings:
  - location: <artifact span>
    type: <error category>
    evidence: <counterexample, check, or derivation>
    request: <minimal requested repair>
```

An accepted artifact has no unresolved findings. When a worker disputes a finding, the evidence and final adjudication remain in the log.

A.3 Additional Results

The final appendix will report per-problem outcomes, confidence intervals, iteration curves, token and tool distributions, Comparator confusion analysis, wiki retrieval audits, and representative trajectories. To avoid selective reporting, the case studies will include successful repairs, false reviewer alarms, false acceptances, and runs that exhaust their budgets.