PAPER FORMALIZATION FOR EFFICIENT AND RIGOROUS PROOF REVIEW

Yuhao Mao[1], **Chenhao Sun**[1], **Yani Zhang**[2], **Martin Vechev**[1]
[1]Department of Computer Science
[2]Department of Information Technology and Electrical Engineering
ETH Zürich, Switzerland
{yuhao.mao, chenhao.sun, martin.vechev}@inf.ethz.ch, yanizhang@mins.ee.ethz.ch

ABSTRACT

Reviewing mathematical proofs is demanding. In our survey of NeurIPS, ICML, and ICLR in 2025, reviewers report checking at least part of a proof for only 28.4% of theoretical publications with proofs. Although missing reports do not establish that proofs go unchecked, this limited reported coverage motivates additional support for rigorous proof review. Proof assistants such as Lean could reduce this burden by automatically checking reasoning, but applying them to frontier proofs presents two challenges. First, complete formalization can unnecessarily require formalizing many results outside the paper. Second, assessing *faithfulness*, the semantic alignment between the formalization and the original mathematics, can require inspecting a large formal body. To address these challenges, we define *Paper Formalization*, propose four metrics for this task, and develop *Paper Formalizer* as a solution. Paper Formalization makes internal arguments machine-checkable while exposing external results as explicit dependencies, allowing each correct paper argument to be formalized without recursively formalizing all dependencies. Paper Formalizer solves the Paper Formalization task while minimizing the review burden. Across theoretical papers spanning multiple topics, Paper Formalizer with GPT-6-Astra uses approximately 90% fewer custom definitions and 64% fewer explicit axioms than the purely agentic baseline. Beyond these reductions, it faithfully formalizes more goals than purely agentic baselines with both GPT-6-Astra and GPT-5.6-Sol. With GPT-6-Astra, it achieves 100% faithfulness, confirmed by two independent human reviewers. Furthermore, in collaborations with authors of three additional publications, Paper Formalizer helps identify and correct theorem-statement typos, ambiguous proof arguments, and incorrect results. After these corrections, it faithfully formalizes all goals in these publications without further human intervention. These results support Paper Formalization as a practical supplement to proof review and Paper Formalizer as a useful tool for authors to improve their proofs, with the potential to make proof review more efficient and rigorous.

1 INTRODUCTION

Reviewing mathematical proofs requires tracing assumptions, checking intermediate arguments, and verifying that cited results apply in the stated setting. This complexity may contribute to limited proof checking in peer review. In our survey of 4,286 theoretical publications accepted at NeurIPS, ICML, and ICLR in 2025, reviewers report checking or attempting to check at least part of a proof for only 28.4% of papers with proofs (Figure 1). Although missing reports do not establish that proofs go unchecked, the limited reported coverage motivates additional support for rigorous proof review to prevent confusion and proof errors from spreading through the community.

Proof assistants such as Lean offer a way to make mathematical reasoning machine-checkable. Recent advances in autoformalization and proof generation use language models to translate informal mathematics and construct formal proofs (Wu et al., 2022; Jiang et al., 2023; Liu et al., 2026). This progress raises a practical question: can formalization support the review of frontier research pa-

pers? Successful compilation alone does not answer this question. A useful formalization must capture the paper's arguments and provide a manageable interface for reviewers to inspect.

Compositional formalization. The first challenge in applying formalization to proof review is that complete formalization can extend far beyond the paper under review. Research proofs routinely invoke cited theorems and standard mathematical properties whose proofs lie outside the paper. Formalizing these dependencies can dominate the formalization effort when they are absent from formal libraries. For example, randomized smoothing combines a Neyman–Pearson argument with Clopper–Pearson confidence bounds to certify robustness (Cohen et al., 2019), but both results are not yet formalized in formal libraries like Mathlib. Formalizing these external dependencies introduces obligations beyond checking its own reasoning. To address this challenge, we define *Paper Formalization*, a task that makes the paper's internal arguments machine-checkable while exposing external results as explicit dependencies. This boundary mirrors the compositional division of responsibility in conventional proof review where cited results are assumed correct, and allows each correct paper argument to be formalized without recursively formalizing all external dependencies. Reviewing the resulting artifact reduces to checking its faithfulness and the reliability of its dependencies. To evaluate Paper Formalization, we propose four complementary metrics: goal coverage, cost, dependency exposure, and inspection complexity. Together, they capture how much of a paper is formalized faithfully, the resources required for formalization, and the human effort required for faithfulness checking.

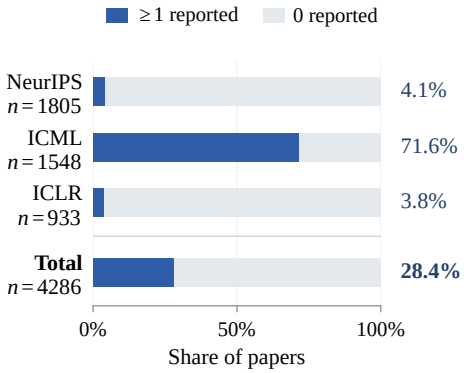


Figure 1: Reported proof-checking rates for top conference publications in 2025. ICML has a relatively high rate because its review form explicitly asks reviewers to report proof checking. The detailed protocol is provided in §A.

Human-readable formalization. The second challenge is the difficulty of checking *faithfulness*, the semantic alignment between a formalization and the original mathematics. While mandatory human review is required to confirm faithfulness, large formalizations can be difficult for human to inspect. Unnecessary axioms, verbose definitions, and duplicate concepts complicate faithfulness checking and limit the benefits of formalization. A formalization should therefore minimize dependency exposure and inspection complexity. To address this challenge, we develop *Paper Formalizer*, solving Paper Formalization while minimizing the review burden. Across nine theoretical papers spanning various topics, Paper Formalizer with GPT-6-Astra reduces custom definitions from 661 to 68 and explicit axioms from 11 to 4, reductions of approximately 90% and 64%, respectively, compared to the purely agentic baseline (§5). Beyond these reductions, it increases the number of faithfully formalized goals from 55 to 60 with GPT-6-Astra and from 29 to 76 with GPT-5.6-Sol. Further, with GPT-6-Astra, Paper Formalizer achieves 100% faithfulness, confirmed by two independent human reviewers.

Paper Formalizer also supports authors in improving their proofs. In collaborations with authors of three additional publications, it helps identify and correct theorem-statement typos, ambiguous proof arguments, and incorrect results. After these corrections, it faithfully formalizes all goals in the three publications without further human intervention. Together, these applications position Paper Formalization as a practical supplement to proof review and Paper Formalizer as a tool for both checking and improving mathematical research.

2 Related Work

Autoformalization and Proof Generation. Autoformalization translates informal mathematics into formal statements checkable by proof assistants such as Lean (Wu et al., 2022; Li et al., 2024). Existing benchmarks span olympiad, undergraduate, and research-level mathematics (Zheng et al., 2022; Azerbayev et al., 2023; Yu et al., 2025; Poiroux et al., 2025; Firsching et al., 2026; Patel et al.,

2026). Proof systems further construct formal proofs for the autoformalized statements (Jiang et al., 2023; First et al., 2023; Yang et al., 2023; Song et al., 2025; Liu et al., 2026). This work focuses on paper-level autoformalization of frontier research, including statements, definitions, and proofs.

Research-Level Autoformalization. For research-level theorem statements, Patel et al. (2023) benchmark translation into Lean syntax while allowing unresolved references to informal definitions. At project scale, Wang et al. (2026) evaluate statement compilation and proof repair on real and convex analysis textbooks and a research paper, while Gloeckle et al. (2026) scale multi-agent formalization to a graduate textbook in algebraic combinatorics. Domain-specific efforts formalize the Vlasov-Maxwell-Landau equilibrium (Ilin, 2026), dimension-8 sphere packing (Hariharan et al., 2026), and neural-network semantics (George et al., 2026). Complete formalization can extend to a target paper's external dependencies. For example, Ju et al. (2026) resolve conjectures and formalize their proofs together with supporting results from external papers, potentially requiring recursive formalization of dependencies. Ren et al. (2026) allow axioms when formalizing three quantum-computation papers, but they also permit blocking subgoals to become assumptions after a fixed attempt limit, so compilation alone does not establish the correctness of the informal statement. In contrast, Paper Formalization defines admissible dependencies by their role in the source paper. Cited results and standard external properties may be assumed with clear references, while internal proof obligations must be discharged. This design supports checking the paper's arguments relative to these dependencies while avoiding the unnecessary burden of formalizing external results.

3 Reducing Proof Review to Faithfulness Checking via Paper Formalization

This section defines the Paper Formalization task and proposes four metrics for evaluating the quality of its solutions, namely goal coverage, cost, dependency exposure, and inspection complexity.

Task definition. Paper Formalization aims to make a paper's mathematical arguments machine-checkable while making their external dependencies explicit. Given an informal paper, the task is to formalize its statements and proofs in Lean or an equivalent proof checker. Cited results and standard mathematical properties outside the paper's arguments may be declared as dependencies, i.e., axioms, in the formalization. Under this formulation, reviewing an informal proof for correctness reduces to checking the faithfulness of its formalization and the reliability of its dependencies. Conceptually, this formulation allows us to formalize every correct research paper. Traditional formalization also requires formalizing the dependencies, which is particularly difficult for many frontier proofs because those dependencies are absent from formal libraries such as Mathlib. For example, randomized smoothing (Cohen et al., 2019) uses a Neyman–Pearson argument to derive robustness guarantees and Clopper–Pearson confidence bounds for Monte Carlo certification. Neither statistical result is formalized in Mathlib v4.34.0. Paper Formalization allows these results to serve as explicit dependencies, avoiding the unnecessary burden of formalizing external dependencies.

This task scope mirrors the division of responsibility in traditional proof review. Reviewers examine the paper's arguments while taking cited results and standard properties as given without requiring self-contained proofs. Paper Formalization makes this boundary explicit by separating the reasoning to be checked from the external results it assumes. Rigorous proof review thus reduces to checking the faithfulness of the formalization and the reliability of its dependencies. Checking faithfulness requires reviewers to understand the formal statements, but need not require mastery of the proof language. Mathematical notation typically makes Lean statements accessible without specialist knowledge of proof assistants, as illustrated by the short tutorial of Dvořák (2025). As we show later, Paper Formalizer, our solution to the Paper Formalization task, handles the most technical aspects and provides minimal reading material, further easing the final human review.

Evaluation criteria. We further propose evaluation criteria for Paper Formalization, illustrated in Figure 2. An explicitly numbered result, such as Theorem 1 or Lemma 2, is an *eligible* (E) informal goal if it has a self-contained proof. A goal is *accepted* (A) if the system identifies no known mathematical issue, such as an incorrect proof step. Acceptance indicates only that the system finds no reasoning errors and does not necessarily establish correctness. For each accepted goal, the system attempts to formalize its statement, dependencies, and proofs in Lean. The goal is *formalized* (FM)

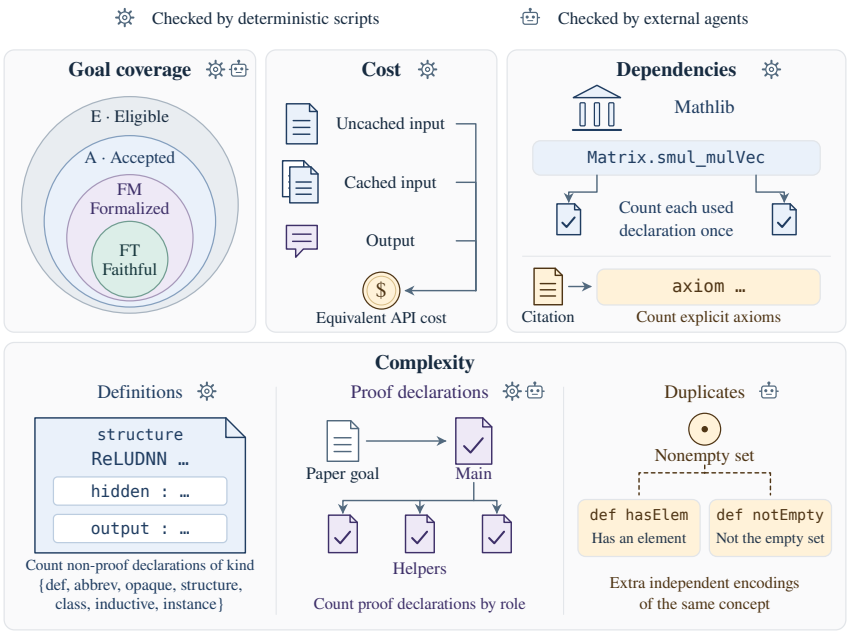


Figure 2: Evaluation metrics for Paper Formalization.

if the final artifact compiles. Since a formalized goal can nevertheless deviate semantically from the original informal goal, we further define formalized goals that also match the original goals semantically to be *faithfully formalized* (FT). Acceptance, completion, and faithfulness are A/E, FM/A, and FT/FM, respectively. Higher completion and faithfulness are desired, while acceptance has no absolute target because unresolved concerns about informal arguments can prevent formalization.

Beyond goal coverage, we evaluate three quality dimensions. First, *cost* measures token consumption by type and the equivalent API cost. Second, *dependency exposure* counts the Mathlib declarations used and the custom axioms. These represent safe dependencies requiring no human checking and unsafe dependencies requiring external validation, respectively. Third, *inspection complexity* consists of three components: (a) definition verbosity counts definition-like declarations, (b) proof-declaration verbosity counts main and helper theorems, and (c) duplication counts definition overlaps representing the same mathematical concept. Lower values are preferable for all metrics in these three dimensions except the number of Mathlib declarations used. This count reflects how effectively the system reuses existing formalizations, with no universal preference for higher or lower values, since they require no human review.

4 Paper Formalizer

This section presents Paper Formalizer, our solution to Paper Formalization. Figure 3 illustrates its workflow. We describe its core design principles in §4.1, the conversion of a paper into a structured contract in §4.2, and formalization from that contract in §4.3.

4.1 Design Principles

Scripted workflow control. Although orchestrator agents are commonly used in agentic systems (Wu et al., 2023; Fourney et al., 2024; Gloeckle et al., 2026), we deliberately use a scripted workflow instead of an orchestrator agent to preserve essential deterministic guarantees. These guarantees, e.g., fixed layout and passed compilation, cannot rely on agent behavior alone, which is intrinsically stochastic. The scripted workflow further handles every deterministic task without agents, e.g., task scheduling and delegation, permission enforcement, result validation, and integration. Agents receive only well-scoped tasks through dedicated delegation. They neither schedule other agents nor approve their own integration.

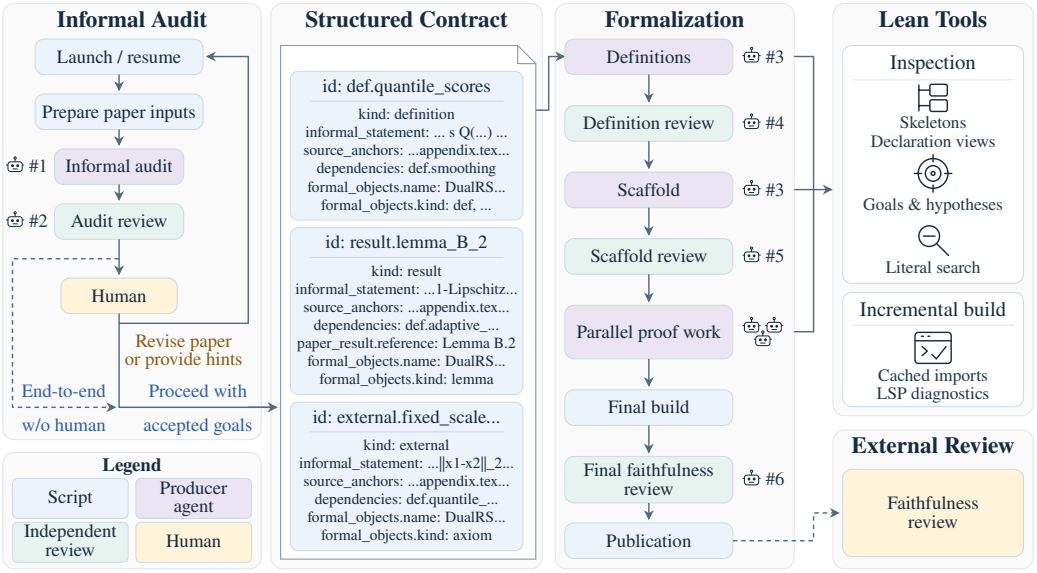


Figure 3: Paper Formalizer workflow. Dashed arrows represent optional paths.

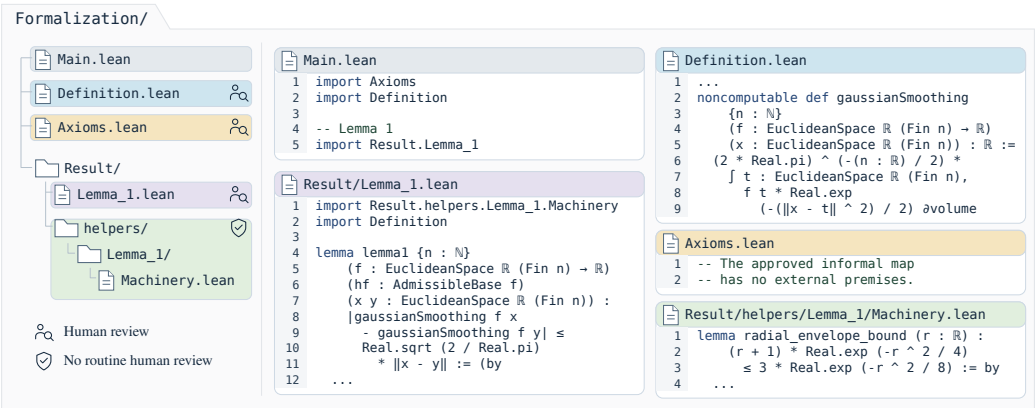


Figure 4: Enforced formalization layout, illustrated with Salman et al. (2019).

Central structured contract. A paper presents its mathematics in free-form text, making it difficult to maintain a consistent specification across workflow stages. We therefore convert the paper into a structured contract in JSON format. This representation allows scripts to prepare scoped assignments and determine task prerequisites, while giving producers and reviewers a shared specification linked to the source. The contract records definitions, goals, external dependencies, and shared helpers, together with their source locations, dependency relations, and intended formal objects. Each informal goal declaration includes a *proof route* describing the paper's reasoning, assumptions, dependency uses, and authorized corrections.

Isolated workspaces and publication. Each agent receives task-specific inputs in an isolated workspace and can write only assigned outputs and temporary files. This separates concurrent edits and preserves shared inputs for validation. Scripts validate outputs before integration. A separate publication stage gather internal artifacts, enforce layout restrictions, and improve statement readability through deterministic, semantically equivalent conversions. The layout separates shared definitions and axioms, result entries grouped by paper reference, and private technical helpers (Figure 4). This is designed to allow human review to focus on axiom reliability and main-result faithfulness. Helper statements and proofs can be skipped since they are internal results required due to formalization semantics and can be safely checked by the compiler alone.

4.2 From Unstructured Paper to Structured Contract

The system takes LaTeX sources, a Lean and Mathlib environment, and optional human guidance as input. During informal audit, an agent inventories eligible goals, reconstructs their effective assumptions, and assesses their arguments and dependencies to prepare the structured contract. An independent reviewer checks the faithfulness of the contract to the original paper. The resulting contract contains the accepted goals and their required dependencies, while a separate inventory retains unaccepted goals and their concerns for human review. This stage screens out goals with readily identifiable concerns or ambiguities to avoid repeated failures or inconsistencies during formalization. By default, the workflow pauses after audit review for a human to inspect concerns, provide clarifications, revise the paper, or authorize formalization with currently accepted goals. When applied end-to-end, the workflow proceeds from audit to formalization without a mandatory human pause. To avoid excluding goals requiring minimal clarifications, e.g., explicitly assuming non-emptiness for a set clearly intended to be non-empty, the system allows source-supported auto-clarifications that preserve the intended setting, boundary cases, parameter regimes, quantitative guarantees, and proof route. The audit review agent also checks these auto-clarifications for faithfulness to the original paper's intent.

4.3 From Contract to Formalization

Formalization begins with an agent implementing the definitions identified in the contract. An independent reviewer checks their faithfulness to the contract before statement formalization to prevent errors from propagating across goals. Once the definitions are approved, the workflow resumes the producer agent for scaffolding, i.e., formalizing theorem statements with `sorry` proof placeholders and the identified external facts. The agent searches Mathlib for existing formalizations of these facts and declares them as axioms when none are found. A separate review then checks the scaffold's faithfulness and the feasibility of the proof routes under the current scaffold. Although these review stages may appear redundant, they detect errors early and prevent their propagation, avoiding wasted effort in later stages.

Once scaffolding is complete, the workflow delegates proof construction to at most three agents in parallel. Each proof agent has write access to only two files: the corresponding result file containing the proof placeholder and an initially empty private helper file. The agent is instructed to retain the informal proof's steps in the result file and place technical lemmas needed for formalization but absent from the informal proof in the helper file. After proof construction, scripts check that the helper file contains only lemmas, with no new definitions or axioms, and that only the proof placeholder is modified in the result file. The proof is merged only after all checks pass.

Lean tools help agents inspect relevant declarations and proof states without repeatedly reading full proof bodies. The `skeleton` operation provides proof-erased views, `declaration` supports exact source inspection, and `search` enables module and declaration lookup. The `goal` operation exposes native proof states and diagnostics, while `build` incrementally compiles assigned outputs and their imports. Proof agents additionally use `check` to validate package scope, statements, proof completeness, and the approved axiom boundary. These tools serve purposes similar to those of Lean LSP MCP (Dressler, 2025), but we implement our own to suit the workflow's needs.

5 Experiments

Setup and baseline. We implement Paper Formalizer's agentic components with Codex and compare it against baseline agents, i.e., Codex sessions with task prompts. Evaluations use GPT-6-Astra with medium reasoning and GPT-5.6-Sol with high reasoning. The baseline agents receive the same environment and permissions as Paper Formalizer, including web search and multi-agent collaboration. They are instructed to complete the same Paper Formalization task, preserving faithfulness to the original paper without requiring external dependencies to be formalized. Their full task prompts are included in §B. In the end-to-end evaluation, all systems operate without human intervention and may raise unresolved concerns to exclude affected goals from formalization. Environment scripts measure cost and count dependencies and definitions, while an external GPT-6-Astra agent with xhigh reasoning evaluates the formalizations using the metrics in §3. Two independent human re-

Table 1: Benchmark results across nine publications on various topics.

Paper	Topic	System	Goals	Cost				Dependencies		Complexity		
			E/A/FM/FT	API ($)	In (M)	Cache (M)	Out (M)	Mathlib	Axioms	Defs	Main/ Helper	Dup.
Kairouz et al. (2015)	Differential privacy	GPT-5.6-Sol	15/3/3/3	15.4	0.64	25.42	0.14	408	0	48	4/27	3
		Ours w/ GPT-5.6-Sol	15/3/3/3	50.3	4.06	64.87	0.41	302	0	8	4/8	0
		GPT-6-Astra	15/6/6/6	149.1	1.73	113.05	0.38	1381	2	177	10/539	11
		Ours w/ GPT-6-Astra	15/5/5/5	22.8	0.77	10.93	0.08	644	0	9	6/9	0
Telgarsky (2015)	Depth separation	GPT-5.6-Sol	6/2/2/2	8.0	0.42	11.29	0.09	526	0	25	2/43	1
		Ours w/ GPT-5.6-Sol	6/6/6/6	19.9	1.18	26.52	0.23	632	0	8	6/21	0
		GPT-6-Astra	6/3/3/3	6.3	0.14	3.31	0.03	388	0	9	3/24	0
		Ours w/ GPT-6-Astra	6/4/4/4	11.4	0.51	4.14	0.04	477	0	8	4/15	0
Yarotsky (2017)	Approximation	GPT-5.6-Sol	11/1/1/1	20.4	0.81	33.95	0.18	635	0	53	2/82	1
		Ours w/ GPT-5.6-Sol	11/10/10/10	288.0	16.85	479.57	1.44	1625	2	19	10/161	3
		GPT-6-Astra	11/4/4/4	69.6	0.71	51.10	0.23	1489	2	122	6/265	8
		Ours w/ GPT-6-Astra	11/10/10/10	216.0	3.91	149.61	0.55	2015	1	7	10/294	0
Arora et al. (2018)	Neural expressivity	GPT-5.6-Sol	20/6/6/4	9.3	0.40	14.59	0.09	442	0	58	8/64	2
		Ours w/ GPT-5.6-Sol	20/8/8/8	64.6	3.18	101.95	0.55	919	1	16	8/97	4
		GPT-6-Astra	20/9/9/9	24.1	0.38	15.14	0.10	812	2	108	10/113	1
		Ours w/ GPT-6-Astra	20/7/7/7	23.6	0.91	9.71	0.10	743	1	17	7/0	0
Zhang et al. (2018)	Tropical geometry	GPT-5.6-Sol	16/6/6/5	11.9	0.56	17.74	0.13	397	1	114	7/71	6
		Ours w/ GPT-5.6-Sol	16/10/10/10	83.7	5.40	126.08	0.58	1307	2	19	12/155	1
		GPT-6-Astra	16/6/6/6	29.7	0.50	18.63	0.12	862	4	101	8/146	2
		Ours w/ GPT-6-Astra	16/6/6/6	29.1	0.93	14.21	0.11	554	1	4	6/50	0
Cohen et al. (2019)	Randomized smoothing	GPT-5.6-Sol	13/5/5/3	17.1	0.73	28.91	0.13	651	5	23	8/36	1
		Ours w/ GPT-5.6-Sol	13/12/12/10	60.7	3.10	97.57	0.46	1510	3	8	13/67	2
		GPT-6-Astra	13/6/6/6	27.4	0.40	19.08	0.09	957	0	20	8/70	3
		Ours w/ GPT-6-Astra	13/4/4/4	20.0	0.72	9.33	0.07	858	0	4	4/26	0
Salman et al. (2019)	Adversarial robustness	GPT-5.6-Sol	2/1/1/1	5.0	0.29	7.03	0.05	416	4	4	1/8	0
		Ours w/ GPT-5.6-Sol	2/2/2/2	53.6	2.62	93.09	0.29	1128	0	2	2/49	0
		GPT-6-Astra	2/0/0/0	2.6	0.10	0.98	0.01	0	0	0	0/0	0
		Ours w/ GPT-6-Astra	2/1/1/1	10.7	0.36	5.31	0.04	807	0	2	1/16	0
Abbe et al. (2021)	Hierarchical learning	GPT-5.6-Sol	48/5/5/1	10.3	0.80	13.21	0.09	203	0	14	5/2	0
		Ours w/ GPT-5.6-Sol	48/12/12/12	79.3	6.68	103.19	0.57	653	0	7	12/43	2
		GPT-6-Astra	48/7/7/6	25.0	0.67	14.92	0.07	473	1	13	7/15	0
		Ours w/ GPT-6-Astra	48/9/9/9	29.9	1.18	13.29	0.10	584	1	11	9/17	0
Baader et al. (2024)	Neural verification	GPT-5.6-Sol	19/11/11/9	17.2	0.63	27.56	0.18	510	0	70	13/80	6
		Ours w/ GPT-5.6-Sol	19/17/17/15	201.0	15.31	285.41	1.28	878	0	12	16/167	3
		GPT-6-Astra	19/15/15/15	40.0	0.48	27.01	0.16	775	0	111	26/178	9
		Ours w/ GPT-6-Astra	19/14/14/14	159.3	2.80	109.43	0.44	971	0	6	14/125	1

Table 2: Aggregate benchmark results across all nine publications.

System	Goals	Goal coverage (%)			Total (per FM)					
	E/A/FM/FT	Acceptance	Completion	Faithfulness	API ($)	Mathlib	Axioms	Defs	Main/Helper	Dup.
GPT-5.6-Sol	150/40/40/29	26.7	**100.0**	72.5	**114.6** (**2.9**)	4188 (104.70)	10 (0.25)	409 (10.23)	50/413 (1.25/10.33)	20 (0.50)
Ours w/ GPT-5.6-Sol	150/80/80/76	**53.3**	**100.0**	**95.0**	901.1 (11.3)	8954 (111.93)	**8** (**0.10**)	**99** (**1.24**)	83/768 (**1.04/9.60**)	**15** (**0.19**)
GPT-6-Astra	150/56/56/55	37.3	**100.0**	98.2	**373.8** (**6.7**)	7137 (127.45)	11 (0.20)	661 (11.80)	78/1350 (1.39/24.11)	34 (0.61)
Ours w/ GPT-6-Astra	150/60/60/60	**40.0**	**100.0**	**100.0**	522.7 (8.7)	7653 (127.55)	**4** (**0.07**)	**68** (**1.13**)	61/552 (**1.02/9.20**)	**1** (**0.02**)

viewers with expertise in Lean and mathematical proofs confirm the 100% faithfulness result reported below for Paper Formalizer with GPT-6-Astra in the end-to-end evaluation.

End-to-end evaluation results. Table 1 reports per-paper results for the nine-paper end-to-end evaluation, while Table 2 summarizes the aggregate results. Paper Formalizer substantially improves acceptance and faithfulness, particularly with the weaker GPT-5.6-Sol: acceptance rises from 26.7% to 53.3%, and faithfulness from 72.5% to 95.0%. Both gains also hold for the stronger GPT-6-Astra, which achieves 100% faithfulness with Paper Formalizer, as confirmed by two independent

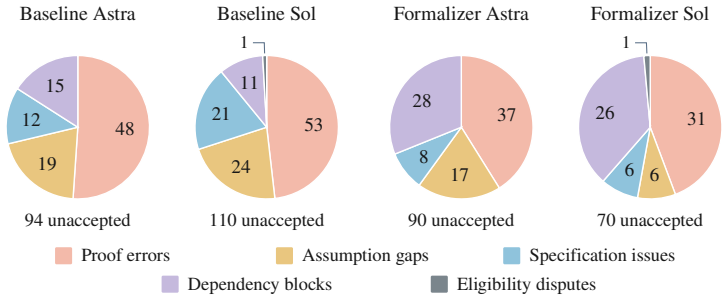


Figure 5: Concerns associated with unaccepted goals across the nine benchmark publications.

human reviewers. Although the GPT-5.6-Sol baseline accepts the fewest goals, Paper Formalizer with GPT-5.6-Sol accepts the most. By comparison, GPT-6-Astra remains more conservative even with Paper Formalizer. The faithfulness improvements are mainly due to targeted multi-stage review mechanisms. Baseline agents also choose to review their formalizations in eight of nine papers with GPT-6-Astra and all nine with GPT-5.6-Sol, but their generic reviews leave some semantic deviations undetected. Notably, the sole unfaithful goal by the GPT-6-Astra baseline occurs in a reviewed formalization. Paper Formalizer's dedicated reviews eliminate observed unfaithfulness with GPT-6-Astra and substantially reduce it with GPT-5.6-Sol. All remaining unfaithful GPT-5.6-Sol goals escape even these dedicated reviews, indicating limitations in the model's ability to detect semantic deviations. Nevertheless, the review process makes Paper Formalizer more costly, especially with the weaker GPT-5.6-Sol, where costs rise from \$2.9 to \$11.3 per FM. This increase mainly reflects multiple rounds of revision due to detected unfaithfulness. With the stronger GPT-6-Astra, the cost overhead for review and revision is only 30%, from \$6.7 to \$8.7 per FM.

Paper Formalizer's most striking advantage is its substantially reduced inspection complexity. While all systems use Mathlib declarations efficiently, Paper Formalizer reduces the axiom exposure from 0.25 to 0.10 per FM for GPT-5.6-Sol and from 0.20 to 0.07 per FM for GPT-6-Astra, reductions of 60% and 65%, respectively. It also reduces the definition verbosity from 10.23 to 1.24 per FM for GPT-5.6-Sol and from 11.80 to 1.13 per FM for GPT-6-Astra, an approximately 90% reduction, and reduces duplicates by 97% with GPT-6-Astra, from 0.61 to 0.02 per FM. Furthermore, Paper Formalizer typically uses a single main theorem declaration per goal, while baseline agents tend to split a goal into multiple main theorems, increasing the effort needed to locate all relevant declarations. Beyond these metric improvements, Paper Formalizer deterministically exposes a minimal, structured interface for human review, as illustrated in Figure 4. Baseline agents produce weakly structured layouts, e.g., creating 70 Lean files to formalize Kairouz et al. (2015).

To understand how systems differ in goal acceptance, we show the concerns raised about unaccepted goals in Figure 5 and the auto-clarifications made in Figure 6. Paper Formalizer reports fewer proof errors, assumption gaps, and specification issues with both models, reflecting more careful judgment, while dependency blocks increase from 15 to 28 with GPT-6-Astra and from 11 to 26 with GPT-5.6-Sol. This is because Paper Formalizer rejects a goal when its proof requires a defective dependency, while baselines tend to use a correct subpart of the dependency to prove dependent goals when the full dependency is defective. These concerns do not establish errors in the original paper unless confirmed by its authors. Among accepted goals, Paper Formalizer makes various auto-clarifications with both models. The stronger GPT-6-Astra baseline exhibits the same behavior, while the weaker GPT-5.6-Sol baseline makes many unfaithful adjustments. These results show that Paper Formalizer avoids aggressively excluding goals requiring only minimal clarification, reducing the need for human clarification. §C further analyzes overlap among accepted goal sets.

Human-in-the-loop evaluation. The previous evaluation leaves identified concerns unaddressed and therefore does not establish their reliability. We further demonstrate that Paper Formalizer raises useful concerns that lead to improvements in the original papers. Working with the authors of three additional anonymized publications, we address concerns identified by Paper Formalizer with GPT-6-Astra. After the authors clarify the concerns and make necessary corrections, Paper Formalizer with GPT-6-Astra faithfully formalizes all 30 goals without further human intervention, achieving 100% acceptance, completion, and faithfulness (Table 3). The original authors confirm the faithfulness of the final formalizations. Their corrections include fixing theorem-statement typos, replacing

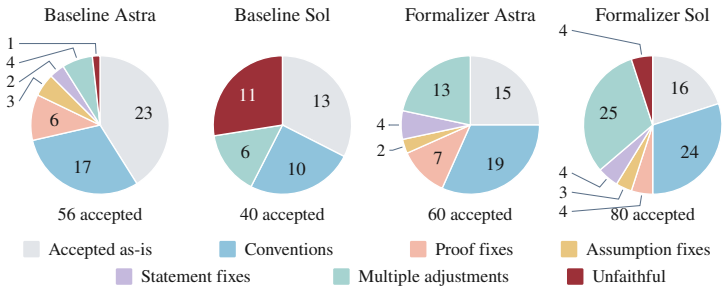


Figure 6: Auto-clarifications made for accepted goals across the nine benchmark publications.

Table 3: Author evaluation on three publications, anonymized for peer review.

Paper	E/A/FM/FT	Goal coverage (%)			Improvements		
		Acceptance	Completion	Faithfulness	Refined Assumptions	Corrected Results	Fixed Proofs
Paper A	12/12/12/12	**100.0**	**100.0**	**100.0**	1	3	1
Paper B	3/3/3/3	**100.0**	**100.0**	**100.0**	0	1	1
Paper C	15/15/15/15	**100.0**	**100.0**	**100.0**	3	4	2
Total	30/30/30/30	**100.0**	**100.0**	**100.0**	4	8	4

ambiguous or erroneous proofs with alternative arguments, and correcting false theorem statements. These results provide further evidence that Paper Formalization, enabled by Paper Formalizer, can substantially improve the rigor of frontier proofs.

6 Discussion

Implications for peer review. As discussed in §3, Paper Formalization can reduce the effort required for rigorous proof review by shifting attention from individual reasoning steps to the faithfulness of the formalization and the reliability of its dependencies. In our evaluation (§5), Paper Formalizer with GPT-6-Astra faithfully formalizes all accepted goals, suggesting the potential to further reduce human effort in faithfulness checking. As an initial step, peer review could encourage or require authors to accompany their proofs with formalizations produced by Paper Formalizer with GPT-6-Astra. Reviewers could cross-check faithfulness themselves or through their own agents. This practice could improve the rigor and efficiency of proof review. Formalization checks the correctness of reasoning, while reviewers still independently assess the significance of the results and the quality of the manuscript.

Implications for LLM training. Paper Formalization could also expand the supply of high-quality Lean training data by pairing research-level informal arguments with faithful, machine-checked proofs. These pairs could support a range of training tasks, including training specialized models such as those developed by Ren et al. (2025). Such models could reduce Paper Formalizer's reliance on advanced general-purpose LLMs.

7 Conclusion

We introduce Paper Formalization, a task that makes a paper's internal arguments machine-checkable while treating external results as explicit dependencies. This formulation reduces proof review to checking faithfulness and dependency reliability. Our implementation, Paper Formalizer, combines scripted workflow control with targeted agent reviews and a compact interface for human inspection. Across nine research papers, it improves acceptance and faithfulness over purely agentic baselines while reducing inspection complexity. With GPT-6-Astra, all accepted goals are faithfully formalized, as confirmed by an independent agent and two independent human reviewers. Collaborations with the authors of three additional papers further demonstrate its value in clarifying arguments and correcting mathematical errors. Together, these results support Paper Formalization as a practical supplement to proof review and Paper Formalizer as a tool for improving research proofs.

BROADER IMPACT

Paper Formalization could improve research reliability by helping authors identify proof errors and making rigorous proof review more accessible. Reusable formalization artifacts could also support subsequent research and the development of mathematical reasoning models. These benefits depend on faithful translation and reliable external dependencies. Treating successful compilation as sufficient evidence of correctness could create false confidence, while treating unsuccessful formalization or unconfirmed agent concerns as evidence of error could unfairly disadvantage authors. Human oversight therefore remains essential when using these artifacts in publication decisions.

Adoption also requires attention to access and confidentiality. Computational costs could disproportionately burden researchers with limited resources, particularly if formalization becomes a submission requirement. Processing unpublished manuscripts with external model services also requires safeguards consistent with authors' consent and venue confidentiality policies.

REPRODUCIBILITY STATEMENT

We provide the formalization artifacts from the end-to-end evaluations, the code, and the Docker environment in the supplementary material. Survey protocol for Figure 1 is included in §A.

LLM DISCLOSURE

This paper was assisted by PIRA (PIRA Project, 2026), a research-assistant agent powered by GPT-5.6-Sol and GPT-6-Astra. The assistance included implementation, writing, result analysis, and figure preparation. The LLMs used in each experiment are documented separately in the corresponding experimental setup. The authors are fully responsible for the presented final content.

REFERENCES

Emmanuel Abbe, Enric Boix-Adsera, Matthew S. Brennan, Guy Bresler, and Dheeraj Nagaraj. The staircase property: How hierarchical structure can guide deep learning. In *Advances in Neural Information Processing Systems*, volume 34, pp. 26989–27002. Curran Associates, Inc., 2021.

Raman Arora, Amitabh Basu, Poorya Mianjy, and Anirbit Mukherjee. Understanding deep neural networks with rectified linear units. In *International Conference on Learning Representations*, 2018.

Zhangir Azerbayev, Bartosz Piotrowski, Hailey Schoelkopf, Edward W. Ayers, Dragomir Radev, and Jeremy Avigad. ProofNet: Autoformalizing and formally proving undergraduate-level mathematics. *arXiv preprint arXiv:2302.12433*, 2023.

Maximilian Baader, Mark Niklas Müller, Yuhao Mao, and Martin Vechev. Expressivity of ReLU-networks under convex relaxations. In *International Conference on Learning Representations*, 2024.

Jeremy Cohen, Elan Rosenfeld, and Zico Kolter. Certified adversarial robustness via randomized smoothing. In *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pp. 1310–1320. PMLR, 2019.

Oliver Dressler. Lean LSP MCP: Tools for agentic interaction with the Lean theorem prover. GitHub repository: oOo0oOo/lean-lsp-mcp, 2025.

Martin Dvořák. How to read Lean. GitHub repository: madvorak/read-lean, 2025.

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.

Emily First, Markus N. Rabe, Talia Ringer, and Yuriy Brun. Baldur: Whole-proof generation and repair with large language models. In *Proceedings of the 31st ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering*, pp. 1229–1241. Association for Computing Machinery, 2023.

Adam Fourney, Gagan Bansal, Hussein Mozannar, Cheng Tan, Eduardo Salinas, Erkang Zhu, Friederike Niedtner, Grace Proebsting, Griffin Bassman, Jack Gerrits, Jacob Alber, Peter Chang, Ricky Loynd, Robert West, Victor Dibia, Ahmed Awadallah, Ece Kamar, Rafah Hosn, and Saleema Amershi. Magentic-One: A generalist multi-agent system for solving complex tasks. *arXiv preprint arXiv:2411.04468*, 2024.

Robert Joseph George, Jennifer Cruden, Will Adkisson, Xiangru Zhong, Huan Zhang, and Anima Anandkumar. TorchLean: Formalizing neural networks in Lean. *arXiv preprint arXiv:2602.22631*, 2026.

Fabian Gloeckle, Ahmad Rammal, Charles Arnal, Remi Munos, Vivien Cabannes, Gabriel Synnaeve, and Amaury Hayat. Automatic textbook formalization. *arXiv preprint arXiv:2604.03071*, 2026.

Sidharth Hariharan, Christopher Birkbeck, Seewoo Lee, Ho Kiu Gareth Ma, Bhavik Mehta, Auguste Poiroux, and Maryna Viazovska. Progress in formalizing sphere packing in dimension 8. *arXiv preprint arXiv:2604.23468*, 2026.

Vasily Ilin. Semi-autonomous formalization of the Vlasov-Maxwell-Landau equilibrium. *arXiv preprint arXiv:2603.15929*, 2026.

Albert Q. Jiang, Sean Welleck, Jin Peng Zhou, Wenda Li, Jiacheng Liu, Mateja Jamnik, Timothée Lacroix, Guillaume Lample, and Yuhuai Wu. Draft, sketch, and prove: Guiding formal theorem provers with informal proofs. In *International Conference on Learning Representations*, 2023.

Haocheng Ju, Guoxiong Gao, Jiedong Jiang, Bin Wu, Zeming Sun, Shurui Liu, Leheng Chen, Yutong Wang, Yuefeng Wang, Zichen Wang, Wanyi He, Peihao Wu, Liang Xiao, Ruochuan Liu, Bryan Dai, and Bin Dong. Automated conjecture resolution with formal verification. *arXiv preprint arXiv:2604.03789*, 2026.

Peter Kairouz, Sewoong Oh, and Pramod Viswanath. The composition theorem for differential privacy. In *Proceedings of the 32nd International Conference on Machine Learning*, volume 37 of *Proceedings of Machine Learning Research*, pp. 1376–1385. PMLR, 2015.

Zenan Li, Yifan Wu, Zhaoyu Li, Xinming Wei, Fan Yang, Xian Zhang, and Xiaoxing Ma. Autoformalize mathematical statements by symbolic equivalence and semantic consistency. In *Advances in Neural Information Processing Systems*, volume 37, pp. 53598–53625. Curran Associates, Inc., 2024.

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. *arXiv preprint arXiv:2601.14027*, 2026.

Nilay Patel, Rahul Saha, and Jeffrey Flanigan. A new approach towards autoformalization. *arXiv preprint arXiv:2310.07957*, 2023.

Nilay Patel, Noah Arias, Davit Babayan, Victoria Cochran, Timothy Libman, Hafsah Mahmood, Liam McCarty, Soli Munoz, Laurel Willey, and Jeffrey Flanigan. MathAtlas: A benchmark for autoformalization in the wild. *arXiv preprint arXiv:2605.14061*, 2026.

PIRA Project. PIRA: A research assistant. `https://github.com/AlgebraLoveme/PIRA`, 2026.

Auguste Poiroux, Antoine Bosselut, and Viktor Kunčak. RLMEval: Evaluating research-level neural theorem proving. In *Findings of the Association for Computational Linguistics: EMNLP 2025*, pp. 10946–10957. Association for Computational Linguistics, 2025.

Yuanjie Ren, Jinzheng Li, and Yidi Qi. MerLean: An agentic framework for autoformalization in quantum computation. *arXiv preprint arXiv:2602.16554*, 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.

Hadi Salman, Greg Yang, Jerry Li, Pengchuan Zhang, Huan Zhang, Ilya Razenshteyn, and Sébastien Bubeck. Provably robust deep learning via adversarially trained smoothed classifiers. In *Advances in Neural Information Processing Systems*, volume 32, pp. 11289–11300. Curran Associates, Inc., 2019.

Peiyang Song, Kaiyu Yang, and Anima Anandkumar. Lean Copilot: Large language models as copilots for theorem proving in Lean. In *Proceedings of the International Conference on Neuro-symbolic Systems*, volume 288 of *Proceedings of Machine Learning Research*, pp. 144–169. PMLR, 2025.

Matus Telgarsky. Representation benefits of deep feedforward networks. *arXiv preprint arXiv:1509.08101*, 2015.

Zichen Wang, Wanli Ma, Zhenyu Ming, Gong Zhang, Kun Yuan, and Zaiwen Wen. M2F: Automated formalization of mathematical literature at scale. *arXiv preprint arXiv:2602.17016*, 2026.

Qingyun Wu, Gagan Bansal, Jieyu Zhang, Yiran Wu, Beibin Li, Erkang Zhu, Li Jiang, Xiaoyun Zhang, Shaokun Zhang, Jiale Liu, Ahmed Hassan Awadallah, Ryen W. White, Doug Burger, and Chi Wang. AutoGen: Enabling next-gen LLM applications via multi-agent conversation. *arXiv preprint arXiv:2308.08155*, 2023.

Yuhuai Wu, Albert Q. Jiang, Wenda Li, Markus N. Rabe, Charles Staats, Mateja Jamnik, and Christian Szegedy. Autoformalization with large language models. In *Advances in Neural Information Processing Systems*, volume 35, pp. 32353–32368. Curran Associates, Inc., 2022.

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. In *Advances in Neural Information Processing Systems*, volume 36, pp. 21573–21612. Curran Associates, Inc., 2023.

Dmitry Yarotsky. Error bounds for approximations with deep ReLU networks. *Neural Networks*, 94:103–114, 2017.

Zhouliang Yu, Ruotian Peng, Keyi Ding, Yizhe Li, Zhongyuan Peng, Minghao Liu, Yifan Zhang, Zheng Yuan, Huajian Xin, Wenhao Huang, Yandong Wen, Ge Zhang, and Weiyang Liu. FormalMATH: Benchmarking formal mathematical reasoning of large language models. *arXiv preprint arXiv:2505.02735*, 2025.

Liwen Zhang, Gregory Naitzat, and Lek-Heng Lim. Tropical geometry of deep neural networks. In *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pp. 5824–5832. PMLR, 2018.

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.

A Survey Protocol for Figure 1

Measurement. Figure 1 measures explicit reviewer reports of checking or attempting to verify part of a detailed mathematical proof, including partial and unsuccessful attempts. These reports do not establish the rigor or completeness of checking. Missing reports do not necessarily imply an absence of checking.

Paper selection. We study accepted main-track papers from NeurIPS, ICML, and ICLR 2025 with public OpenReview records matched to official proceedings, excluding workshops, position papers,

and the NeurIPS Datasets and Benchmarks track. Screening uses case-insensitive regular expressions on text extracted from final manuscripts and linked supplementary PDFs. Eligibility requires a line beginning with theorem, lemma, proposition, corollary, or claim followed by a numeric or letter-based identifier, together with a proof heading and a nonempty body. We exclude table-of-contents entries and proof passages marked as sketches, outlines, omitted or deferred proofs, or references elsewhere. The final sample sizes n in Figure 1 are 1,805 for NeurIPS, 1,548 for ICML, and 933 for ICLR, totaling 4,286 papers. This selection process is intentionally conservative.

Classification. We use the latest public versions of official reviews, excluding author responses, meta-reviews, decisions, and follow-up discussions. After retaining narrative fields and removing template text, we select sentences containing keywords such as proof, theorem, lemma, proposition, or corollary, including plurals and common abbreviations. These sentences serve as input to `gpt-5.6-luna` with low reasoning effort in a fresh agent session for each paper. The model identifies whether any sentence suggests checking any part of a proof during review. The numbers of papers with at least one review reporting proof checking are 74 for NeurIPS, 1,108 for ICML, and 35 for ICLR, totaling 1,217 papers.

B Baseline Instructions

The baseline receives the following launch prompt, with parameters substituted at runtime.

```
Set this as the active persistent goal and keep working until it is complete:

Formalize every informal goal in ${arxiv_url} in LEAN4.

An informal goal is an explicitly stated theorem, proposition, lemma, corollary, or
comparable mathematical result with its own explicit numbering in the original paper
and an explicit paper-provided proof argument in the main text, an appendix, or
supplementary proofs.  Section numbers, equation numbers, and agent-invented labels do
not count as result numbering.  Unnumbered claims, unproved assertions, conjectures,
and empirical observations are not informal goals.  A cited result qualifies only if
it meets these criteria and the paper itself supplies a self-contained proof; a
citation to an external proof is insufficient.  Definitions and supporting helpers are
not additional informal goals.

You are allowed to websearch and use subagents properly.  You are allowed to write an
escalation.md to explain concrete cases where formalization is impossible.  Cited
results and cited properties that are not proved in the paper may be accepted as
axioms; make their sources, precise statements, and applicability explicit.  Do not
use axioms to replace the paper's own results or arguments.

Be faithful to the paper's argument; do not invent new proofs or silently fix proofs.
Preserve the original statements, assumptions, and scope.  Identify each informal goal
by its original reference, and report unresolved gaps rather than silently changing or
omitting affected goals.

Keep all writable work and final artifacts under ${workspace_root}.  The matching Lean
4/Mathlib cache is mounted read-only at ${mathlib_root} and is available for use.

Include the following completion criterion in the saved goal.  Mark the goal complete
only when every informal goal is either formalized or escalated due to paper
source-level problems.  Implementation difficulties or usage limits do not count as
paper source-level problems.

Wait for all child agents to finish before completing the goal or exiting.
```

C Accepted Goal Overlap

The systems accept overlapping but distinct goal sets. Of 150 eligible goals, 90 are accepted by at least one system, 32 by all four, and 60 by none. To account for unequal acceptance counts, Figure 7 uses the overlap coefficient $|A_i \cap A_j| / \min(|A_i|, |A_j|)$, where A_i is system i's accepted set pooled across all nine papers. This measures the fraction of the smaller set contained in the larger. A value of 100% indicates containment and does not require identical accepted sets.

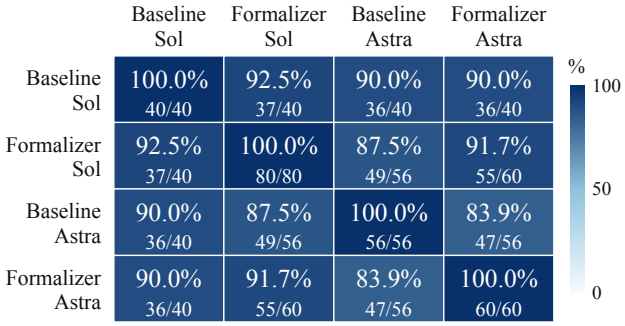


Figure 7: Pairwise accepted-goal overlap pooled across nine papers. Each cell shows the overlap coefficient as a percentage and the exact ratio of shared goals to the smaller accepted-set size. Color uses a 0–100% scale.

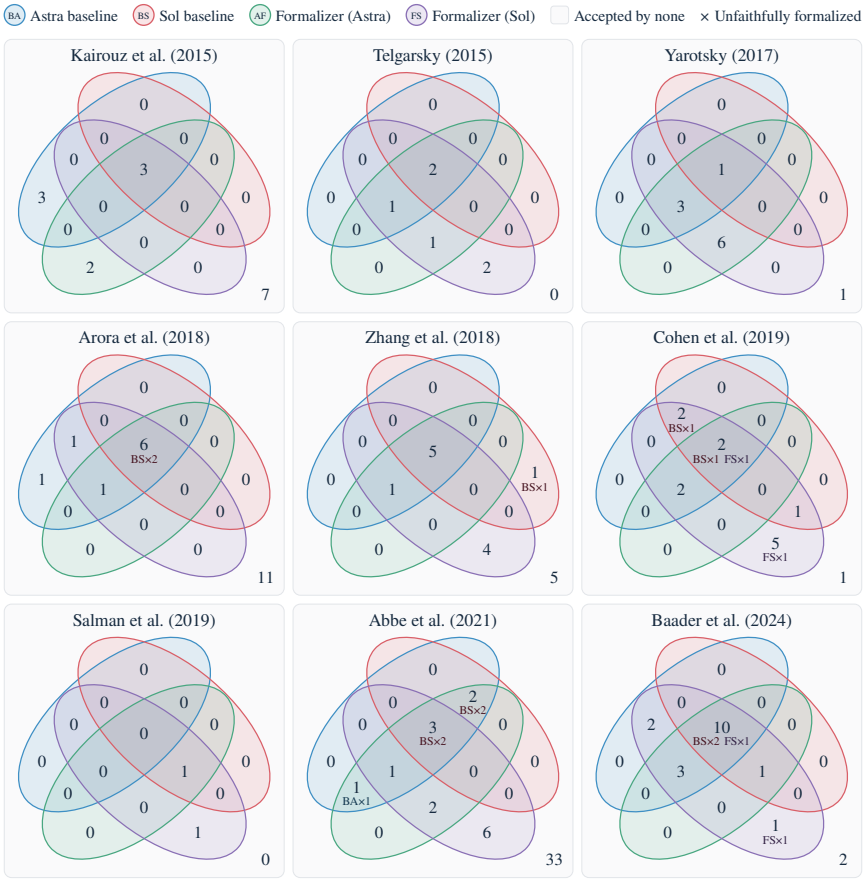


Figure 8: Analysis of goals accepted for each paper. BS, BA, and FS denote the Sol baseline, Astra baseline, and Paper Formalizer with Sol, respectively.

Pairwise overlap ranges from 83.9% to 92.5%. Paper Formalizer retains 37 of the 40 baseline goals with GPT-5.6-Sol (92.5%) and 47 of the 56 baseline goals with GPT-6-Astra (83.9%). The two Paper Formalizer configurations share 55 of the smaller set's 60 goals (91.7%). Thus, their accepted sets overlap substantially despite differences in total acceptance.

High aggregate overlap can coexist with local differences. For Kairouz et al. (2015), the GPT-6-Astra baseline and Paper Formalizer accept six and five goals, respectively, but share only three, yielding 60% overlap (Figure 8). The aggregate matrix therefore complements the per-paper analysis. Neither overlap nor acceptance alone establishes correctness or faithfulness.