FORMALMATHARENA FOR RELIABLE EVALUATION FROM NATURAL LANGUAGE PROBLEMS TO LEAN PROOFS

Anonymous authors
Paper under double-blind review

ABSTRACT

Practical formal theorem proving inherently entails two interdependent stages: translating natural-language problems into formal statements and constructing machine-checked proofs. Yet, recent advances in LLM-based theorem proving largely rely on proof-only benchmarks or decoupled evaluations of formalization and proof quality. The former neglects the critical formalization step of real-world applications while the latter severs the interplay between the two tasks, both potentially mischaracterizing the formal theorem-proving capabilities of LLMs. To address these limitations, we introduce FormalMathArena, an agentic workflow for evaluating the entire formal theorem-proving process. FormalMathArena integrates semantic alignment, proof-assistant verification, and proof-guided failure diagnosis to disentangle and independently assess formalization fidelity, proof validity, and overall end-to-end success. We further construct a benchmark of 5,427 mathematical problems and validate FormalMathArena on a dedicated expert-annotated subset, confirming its high diagnostic reliability. Applying our framework to representative LLMs at scale reveals a performance gap of up to 27.39 percentage points between proof accuracy and end-to-end accuracy, identifying accurate and proof-ready formalization as a major bottleneck. Beyond evaluation, repurposing FormalMathArena as an action-verification harness improves end-to-end accuracy by up to 22.5 percentage points relative to the initial outputs on an additional held-out subset. These results establish FormalMathArena as both a reliable evaluation framework and an effective inference-time verifier for end-to-end formal theorem proving. *Our code and data will be released upon acceptance.*

1 INTRODUCTION

Real-world formal theorem proving fundamentally comprises two interdependent stages: translating natural language problems into formal statements (problem formalization) and constructing reasoning in proof assistant's language (proof generation). Large language models (LLMs) are frequently reported to have achieved impressive milestones in this domain, showing strong performance on established benchmarks. Many of these results are obtained in a "proof-only" setting, where formal statements are provided in advance and models are evaluated solely on proof construction (Zheng et al., 2022; Yang et al., 2023; Yu et al., 2025b). This setting does not directly measure whether a model can formalize a natural-language problem and construct a proof of the resulting statement (Yu et al., 2025a; Ospanov et al., 2025).

The proof-only setting has been widely adopted primarily because evaluating the entire pipeline, particularly problem formalization, is exceptionally challenging. Evaluating problem formalization is fundamentally multifaceted. On one hand, it requires assessing intrinsic semantic fidelity, ensuring that the formal statement aligns strictly with the original problem without introducing extraneous axioms that permit unintended proof shortcuts. On the other hand, it requires evaluating its extrinsic adequacy for proof construction: even if a formalization faithfully captures the problem, it may be overly complex or lack implicit external conditions necessary to support the proof, rendering downstream proving intractable. These difficulties far exceed the complexity of proof-only evaluation, which relies straightforwardly on automated, deterministic checking via proof assistants such as Lean de Moura & Ullrich (2021). Several studies evaluate statement autoformalization through

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.

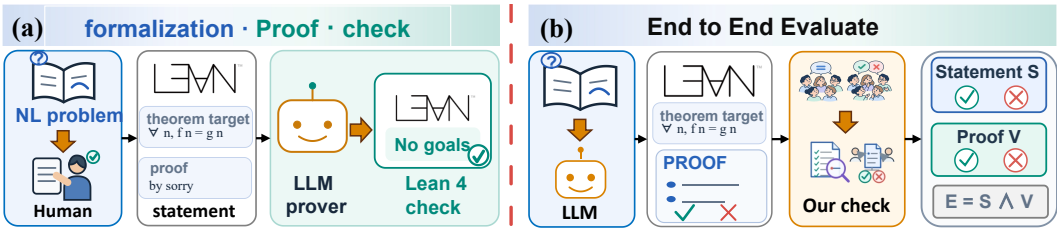

Figure 1: Proof-only and end-to-end evaluation. (a) An LLM generates a proof for a human-provided formal statement. (b) FormalMathArena evaluates semantic alignment and proof validity for a statement and proof generated from a natural-language problem.

dedicated semantic metrics (Azerbayev et al., 2023; Poiroux et al., 2025; Zhang et al., 2026), but these metrics do not by themselves identify why a subsequent proof attempt fails. For example, when an incomplete or overly complex formalization leads to proof failure, such decoupled evaluations misattribute the error, potentially overrating the flawed statement while unfairly penalizing the prover. To address these challenges, we introduce FormalMathArena, a novel framework for evaluating end-to-end formal theorem proving.

FormalMathArena adopts a modular-to-holistic evaluation strategy. It first independently assesses the intrinsic semantic fidelity of problem formalization and the correctness of the proof, and then employs abductive reasoning to categorize proof failures, thereby evaluating extrinsic proof adequacy. Because evaluating semantic fidelity and diagnosing proof failures both require sophisticated semantic understanding, FormalMathArena develops dedicated LLM-based agentic workflows for these tasks. Specifically, the Semantic Alignment (SA) module assesses semantic fidelity via a fast-and-slow design with KV-cache reuse Zheng et al. (2024), performing this challenging assessment both efficiently and accurately. Meanwhile, the Proof-Guided Diagnosis (PGD) module attributes proof failures and applies a repropose-and-verify mechanism to substantiate its diagnoses, ensuring attribution reliability. Relying on these modules, FormalMathArena provides a pair of modified metrics for problem formalization and proof checking, enabling a more precise and reliable assessment of LLM formal theorem proving capabilities.

To ground our evaluation, we construct a benchmark of 5,427 mathematical problems spanning diverse difficulty levels. We first confirm the reliability of FormalMathArena through a meta-evaluation on 200 representative problems across diverse LLMs, where it achieves 91.50% accuracy in semantic fidelity assessment and 86.00% in proof failure attribution compared to human-expert judgments. With this reliability established, we deploy FormalMathArena across 4,827 problems to systematically evaluate ten representative open- and closed-source LLMs. Our findings reveal that formalization errors are an important source of end-to-end failure, with proof accuracy exceeding end-to-end accuracy by up to 27.39 percentage points. Strong performance on proof-only benchmarks does not necessarily translate into strong end-to-end performance, because these benchmarks assume that the problem has already been correctly formalized. Crucially, FormalMathArena extends beyond evaluation: repurposed as an inference-time verifier under the harness-as-action-verifier paradigm (Lou et al., 2026), it elevates cumulative end-to-end success on 400 test problems from 39.1% to 52.1%. These findings establish FormalMathArena both as a reliable diagnostic evaluator and as an effective test-time harness for advancing formal mathematical reasoning. Our contributions are threefold:

1. We introduce FormalMathArena, an agentic framework that integrates semantic fidelity checking and failure attribution to reliably evaluate the entire formal proving process for LLMs.
2. We construct a benchmark and evaluate representative LLMs on it. Results obtained with FormalMathArena reveal differences in formalization and proof performance across models and identify formalization errors as an important source of end-to-end failure.
3. Extensive experiments show FormalMathArena as an effective action-verification harness that improves end-to-end accuracy, demonstrating its value for reliable evaluation and inference-time verification.

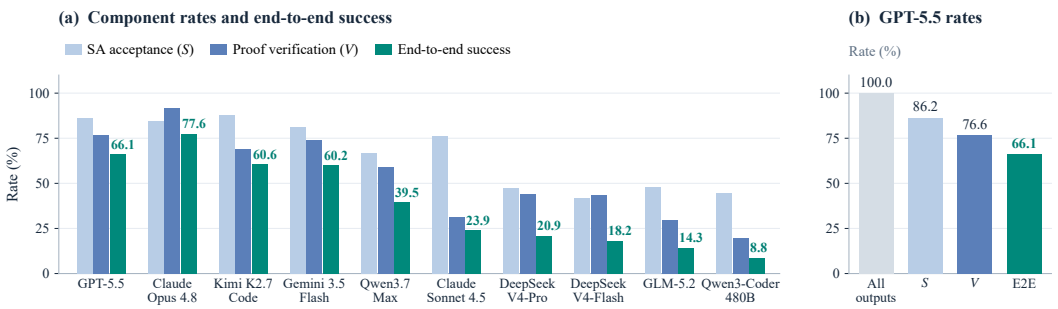

Figure 2: SA acceptance, conditional proof verification, and end-to-end success rates for (a) ten models and (b) GPT-5.5. SA (S) and E2E rates use all 4,827 problems; verification (V) rates use only outputs that pass SA. The 100% reference in (b) represents all outputs.

2 Related Work

2.1 Formal Proving

Established formal theorem-proving benchmarks, including miniF2F, LeanDojo, FormalMATH, and PutnamBench, provide formal statements in advance and evaluate whether generated proofs are accepted by a proof assistant (Zheng et al., 2022; Yang et al., 2023; Yu et al., 2025b; Tsoukalas et al., 2024). Recent approaches improve proof generation through mathematical pretraining and synthetic proof data (Azerbayev et al., 2024; Xin et al., 2024; Lin et al., 2025), reinforcement learning (Ren et al., 2025; Wang et al., 2025; Hubert et al., 2026), and retrieval, proof search, and iterative refinement with proof-assistant feedback (Yang et al., 2023; Ma et al., 2026; Jiang et al., 2023; Wu et al., 2024). Lean Eval similarly evaluates submitted proofs against trusted problem formalizations supplied by the evaluator (Lean Project, 2026), while FATE extends formal theorem-proving benchmarks to abstract and commutative algebra across multiple difficulty levels (Jiang et al., 2026). However, because the formal statement is fixed, results in this setting do not measure whether a model can correctly formalize the original natural-language problem.

2.2 Autoformalization

Autoformalization translates natural-language mathematical problems into machine-checkable statements (Wu et al., 2022). Datasets such as ProofNet and Lean Workbook provide paired natural-language problems and Lean formalizations for developing and evaluating such systems (Azerbayev et al., 2023; Ying et al., 2024). Since successful compilation does not guarantee semantic faithfulness, existing work evaluates generated statements through back-translation, equivalence checking, and expert annotation (Ying et al., 2024; Poiroux et al., 2025). FormalAlign learns an evaluator for alignment between natural-language and formal statements (Lu et al., 2025). Beyond Compilation combines Lean compilation with cross-model semantic judgments and validates its evaluations against human annotations (Zhang et al., 2026). FormalMATH uses multi-LLM semantic verification and negation-based disproof filtering during benchmark construction (Yu et al., 2025b). More recent work constructs semantic probes from the original problem and applies formal entailment checks to detect semantic changes in generated formalizations (Mohammad & Sheikh, 2026).

These approaches primarily evaluate the generated formal statement itself. FormalMathArena additionally considers the downstream proof attempt and its verification outcome.

2.3 End-to-End Formal Proving

Recent work has begun to evaluate complete pipelines from natural-language problems to formal proofs. Mathesis combines autoformalization, formalization evaluation, and theorem proving (Yu et al., 2025a), while the revised miniF2F benchmark evaluates pipelines that first generate a formal statement and then construct its proof (Ospanov et al., 2025). Related studies further analyze cases in which formally verified proofs rely on unfaithful formalizations or unintended assumptions (Kim et al., 2026).

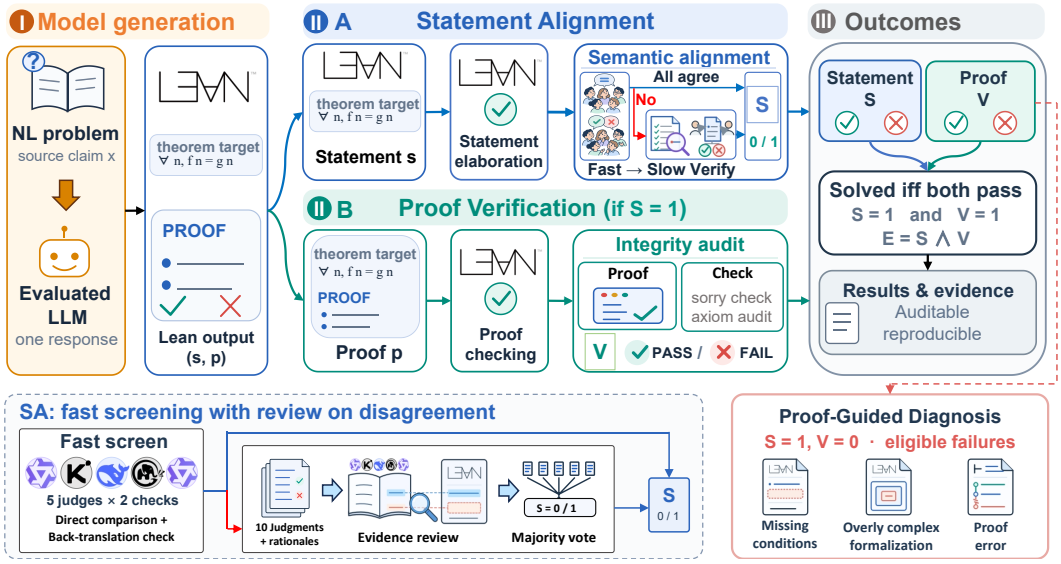

Figure 3: FormalMathArena evaluation pipeline. Semantic alignment (SA) combines semantic judgments with Lean checks of the generated code. For outputs that pass SA, Lean 4 verifies the proof. When proof verification fails, proof-guided diagnosis (PGD) checks whether the failure is caused by the formalization or the proof.

FormalMathArena complements these efforts by focusing on *automated failure attribution*. It combines semantic alignment, proof verification, and proof-guided diagnosis to distinguish failures caused by formalization from those caused by proof generation. This decomposition enables separate measurement of formalization accuracy, proof accuracy, and end-to-end accuracy. The resulting diagnostic feedback can also support inference-time revision under the harness-as-action-verifier paradigm (Lou et al., 2026).

3 FormalMathArena

Given a natural-language problem x, the LLM under evaluation generates a formal statement s and a candidate proof p (Figure 3). FormalMathArena first uses its Semantic Alignment (SA) module to assess semantic fidelity and apply Lean checks to the generated code. If the output passes SA, Lean 4 verifies whether p is a valid proof of s. If verification fails, the Proof-Guided Diagnosis (PGD) module assesses whether the failure comes from the proof itself or from the way the problem was formalized. The alignment decision, Lean verification result, and PGD diagnosis are then combined to compute three metrics: formalization accuracy, proof accuracy, and end-to-end accuracy.

3.1 Semantic Alignment

The Semantic Alignment (SA) module employs five jurors from multiple LLM families within a fast-and-slow workflow. The fast stage collects initial judgments, while the slow stage deliberates on cases with disagreement.

To ensure reliability, each juror evaluates whether the formal statement s faithfully aligns with the original problem x through two complementary perspectives: directly assessing the correspondence between s and x, and back-translating s into natural language $\hat{x}$ for semantic consistency checking (full prompts are provided in the supplementary material). Under this structured prompting, each juror produces a verdict followed by an elicited chain-of-thought rationale, yielding two independent judgments per juror and ten judgments in total.

In the fast stage, we emulate intuitive System 1 reasoning (Kahneman, 2011). Each juror directly outputs its verdict while bypassing chain-of-thought generation, and its key-value (KV) cache is preserved for potential reuse (Zheng et al., 2024). To accelerate inference, we apply vocabulary

masking to compute probabilities exclusively over the candidate tokens "Yes" and "No" rather than the entire vocabulary, substantially cutting computational overhead. If all ten judgments reach consensus, SA adopts this unanimous verdict as its semantic judgment.

In the slow stage, we emulate deliberative System 2 reasoning (Kahneman, 2011). If the ten judgments are not unanimous, SA restores the stored KV caches to resume decoding without recomputing context, generating the accompanying chain-of-thought explanation for each judgment. The five models then independently inspect the complete panel of ten judgments and their rationales. A majority vote across their five final decisions establishes the semantic judgment.

In addition to semantic consensus, the formal statement s is verified independently in the proof assistant in isolation from any candidate proof. The final decision is set to $S = 1$ only when s passes this standalone statement check and receives a positive semantic judgment; otherwise, $S = 0$.

3.2 Proof Verification

For each output that passes SA ($S = 1$), Lean 4 checks whether p is a valid proof of s, using the generated code without modification. The verification result is $V = 1$ if the proof is accepted, and $V = 0$ otherwise.

3.3 Proof-Guided Diagnosis

For outputs that pass SA ($S = 1$) but fail proof verification ($V = 0$), PGD assesses whether the failure comes from the formalization or the proof. To make this distinction, it uses a single LLM to jointly examine (x, s, p). Formalization problems are further divided into two types: missing assumptions or conditions and excessive complexity.

PGD first checks whether the reasoning in p relies on an assumption or condition of x that s omits. If so, it assigns the missing-conditions label $q_{\mathrm{M}} = 1$.

If no omission is identified, PGD examines whether the structure of s makes the attempted proof unnecessarily difficult. If no such structure is identified, it assigns the proof-error label $q_{\mathrm{P}} = 1$. Otherwise, the model proposes a more direct formalization s' of the same problem and generates a candidate proof p'. If p' passes Lean 4 verification, PGD assigns the overly complex label $q_{\mathrm{C}} = 1$; otherwise, it assigns $q_{\mathrm{P}} = 1$. The resulting diagnosis is represented by

$$(q_{\mathrm{M}}, q_{\mathrm{C}}, q_{\mathrm{P}}) \in \{0, 1\}^3, \qquad q_{\mathrm{M}} + q_{\mathrm{C}} + q_{\mathrm{P}} = 1.$$

3.4 Evaluation Metrics

In this subsection, S is the final SA pass rate over all outputs, and V is the proof verification rate among outputs that pass SA. Let q_{P} be the proportion of SA-passing verification failures that PGD classifies as proof errors. Its denominator includes all outputs with $S = 1$ and $V = 0$.

Building on these component outputs, we define three derived accuracy metrics:

Formalization Accuracy is computed as

$$\mathrm{Acc}_{\mathrm{form}} = S\big[V + (1 - V)q_{\mathrm{P}}\big] \in [0, 1], \tag{1}$$

which reflects the quality of problem formalization, accounting for both semantic accuracy and whether the formalization is appropriate for supporting subsequent proof construction. A value of 0 indicates the poorest formalization quality, whereas a value of 1 indicates that all problems are accurately and appropriately formalized.

Proof Accuracy is derived as

$$\mathrm{Acc}_{\mathrm{proof}} = \frac{V}{V + (1 - V)q_{\mathrm{P}}} \in [0, 1]. \tag{2}$$

It combines Lean 4 verification with PGD's attribution of failed proofs and excludes failures caused by problem formalization, thereby reflecting the actual quality of proof generation. A value of 0 indicates the poorest proof-generation performance, whereas a value of 1 indicates that all proofs generated from acceptable formalizations are valid.

Table 1: Performance of ten models on 4,827 problems. The SA pass rate (S) uses all problems; proof verification (V) is conditional on passing SA. Derived accuracies are defined in Section 3.4. Boldface indicates the highest value in each column.

	Component pass rate (%)			Derived accuracy (%)		
Model	Statement elab.	SA pass S	Proof verif. V	Form. accuracy	Proof accuracy	E2E accuracy
GPT-5.5	93.35	86.24	76.60	85.10	77.63	66.07
Claude Opus 4.8	93.20	84.79	**91.50**	84.71	**91.59**	**77.58**
Kimi K2.7-Code	**93.70**	**88.09**	68.84	**86.84**	69.82	60.64
Gemini 3.5 Flash	89.85	81.09	74.22	80.11	75.12	60.18
Qwen3.7-Max	86.89	66.92	59.04	65.76	60.08	39.51
Claude Sonnet 4.5	87.32	76.47	31.27	72.94	32.77	23.91
DeepSeek V4-Pro	84.92	47.40	44.19	45.49	46.04	20.94
DeepSeek V4-Flash	81.38	41.58	43.70	39.88	45.56	18.17
GLM-5.2	80.13	47.81	29.85	46.47	30.72	14.27
Qwen3-Coder 480B	69.73	44.35	19.94	41.52	21.31	8.85

End-to-End Accuracy is computed as follows:

$$\mathrm{Acc}_{\mathrm{e2e}} = SV \in [0, 1]. \tag{3}$$

This score reflects the overall level of success of the end-to-end formal theorem-proving process. A value of 0 indicates that no problems are both correctly formalized and successfully proved, whereas a value of 1 indicates that all problems are accurately formalized and successfully proved.

These metrics satisfy the following relation:

$$\mathrm{Acc}_{\mathrm{e2e}} = \mathrm{Acc}_{\mathrm{form}} \cdot \mathrm{Acc}_{\mathrm{proof}}. \tag{4}$$

This decomposition enables independent diagnosis and improvement of formalization and proof capabilities: a low end-to-end score can be traced to whichever factor is the binding constraint. Detailed derivations and insights are provided in the supplementary material.

3.5 FormalMathArena as an Action Verifier

FormalMathArena can also be integrated as an effective verifier into the harness-as-action-verifier framework (Lou et al., 2026) with minor modifications. SA and PGD provide their decisions and any explanations generated during evaluation. The binary outputs S and V are used as the corresponding `is_legal_action()` signals (Lou et al., 2026), determining whether a generated formalization or proof should be accepted. When SA rejects an output ($S = 0$), the rejection and any available semantic explanation or Lean diagnostics are returned to the prover for revision. When Lean rejects the proof ($V = 0$) and PGD attributes the failure to the formalization ($q_{\mathrm{M}} = 1$ or $q_{\mathrm{C}} = 1$), the corresponding diagnosis (missing assumptions vs. overly complex encoding) is similarly fed back. Each revision cycle checks SA before proof verification, and a maximum of three rounds is permitted. This structured feedback replaces blind trial-and-error with diagnosis-driven revision (evaluated in *Inference-Time Verification and Revision*).

4 Experiments

We first establishcompare a meta-evaluation to validate SA's semantic judgments and PGD diagnoses with expert annotations. We then evaluate the formalization, proof, and end-to-end accuracy of ten LLMs and analyze their failures. Finally, we test whether FormalMathArena feedback improves end-to-end success during revision and examine how the gains vary across models.

4.1 Experimental Setup

The benchmark contains 5,427 problems drawn from public sources, including a cleaned NuminaMath-RL derivative (LLMTeamAkiyama, 2025), Lean Workbook (Ying et al., 2024),

Table 2: Expert validation of SA and PGD modules against human expert labels. Metrics are reported as percentages, with F1 macro-averaged. n denotes the number of annotated evaluation cases.

Overall agreement				
Module		n	Acc.	F1
SA		200	91.50	91.50
PGD		200	86.00	86.24
Class-specific metrics				
Module	Class	n	Prec.	Recall
SA	Misaligned	102	93.81	89.22
PGD	Proof error	70	78.38	82.86
PGD	Missing conditions	70	95.08	82.86
PGD	Overly complex	60	86.15	93.33

MATH (Hendrycks et al., 2021), FormL4/PDA (Lu et al., 2024), and GSM8K (Cobbe et al., 2021). We remove duplicates and divide the problems into three nonoverlapping sets: 4,827 for the main evaluation, 200 for expert validation, and 400 for the revision study. In the main study, we evaluate ten LLMs using the same generation prompt and evaluation protocol. Each model generates one formal statement and one candidate proof per problem.

The familiesy of SA's jursors areare : GLM-5.1, Qwen3-Max, Kimi K2.6, DeepSeek-V3.2, and Qwen3-Coder 480B, respectively. PGD uses GPT-5.5 as its diagnostic model. We run all Lean checks with Lean 4.21 and the same fixed Mathlib environment. The supplementary material describes the generation and evaluation settings.

4.2 Meta-Evaluation

Using the 200-problem validation set, we construct separate expert-annotated sets of 200 cases for SA and PGD. For SA, we randomly sample one model to tackle each problem to eliminate the model biases. Then, for each model inference results, experts take the model inferred formal problem statements and the original problem to label the formal statements as semantically aligned or misaligned. Results are shown in Table 2. SA's semantic judgments agree with the expert labels on 183 of 200 cases, achieving an accuracy and macro-F1 of 91.50%. For misaligned statements, SA achieves a precision of 93.81% and a recall of 89.22%.

The PGD validation set contains 70 proof-error cases, 70 cases with missing assumptions or conditions, and 60 cases with overly complex formalizations. PGD correctly classifies 172 of the 200 cases, achieving an accuracy of 86.00% and a macro-F1 of 86.24%. Table 2 summarizes the results for both modules; the confusion matrices are provided in the supplementary material. All these results demonstrate the effectiveness of the LLM parts in FormalMathArena, proving its reliability is satisfactory for evaluation of formal theorem proving.

4.3 Benchmarking Representative LLMs on End-to-End Proving

Table 1 presents the evaluation results across 4,827 benchmark problems. The first three columns report intermediate component pass rates computed by FormalMathArena during execution, while the right three columns show the derived formalization, proof, and end-to-end (E2E) accuracies calculated on the basis of these stages (Section 3.4). In terms of aggregate statistics, Claude Opus 4.8 leads with an E2E accuracy of 77.58%, followed by GPT-5.5 at 66.07%. Furthermore, models exhibit clear stage-level trade-offs: Kimi K2.7-Code achieves the highest formalization accuracy (86.84%) but is limited by proof synthesis (69.82%), whereas Claude Opus 4.8 relies on superior proof accuracy (91.59%) to offset its lower formalization rate (84.71%) and secure the top rank.

Analyzing performance across models confirms that problem formalization represents the primary practical bottleneck. Formalization failures effectively truncate the pipeline before proof verification can succeed, causing end-to-end accuracy to fall substantially below conditional proof accuracy across all models. Across all ten evaluated models, this gap averages 16.05 percentage points and

Table 3: Paired revision on 400 problems per model. B_0 is the initial success rate; L and T are cumulative rates after up to three revision rounds with Lean-only and FormalMathArena feedback, respectively. Gains $T - L$ are computed before rounding and reported in percentage points (pp). For each model, the better feedback result is bold.

	End-to-end success (%)			Gain (pp)
Model	Initial B_0	Lean L	FMA T	$T - L$
Claude Opus 4.8	70.5	73.0	**77.5**	+4.5
GPT-5.5	65.5	82.0	**82.8**	+0.8
Gemini 3.5 Flash	63.0	69.2	**69.5**	+0.2
Kimi K2.7-Code	57.5	70.5	**74.0**	+3.5
Qwen3.7-Max	37.8	**47.2**	46.8	−0.5
Claude Sonnet 4.5	26.2	45.5	**46.8**	+1.2
DeepSeek V4-Pro	20.5	30.8	**43.0**	+12.2
DeepSeek V4-Flash	20.0	27.2	**34.8**	+7.5
GLM-5.2	18.8	29.5	**32.0**	+2.5
Qwen3-Coder 480B	11.5	**17.0**	13.8	−3.2
Pooled	**39.1**	**49.2**	**52.1**	**+2.9**

reaches up to 27.39 percentage points for DeepSeek V4-Flash, whose proof accuracy attains 45.56% while end-to-end accuracy drops to 18.17%. This stark divergence confirms that conventional proof-only benchmarks substantially overestimate real-world mathematical capabilities by bypassing the severe attrition during formal translation.

4.4 Failure Analysis

To pinpoint the exact root causes of pipeline failures and determine which stage constitutes the primary practical obstacle, we conduct a fine-grained failure analysis across all 48,270 submissions generated by the ten representative models on the 4,827 benchmark problems (Table 4. In total, 18,831 submissions (39.01%) achieve end-to-end success, leaving 29,439 unsuccessful outputs partitioned into 16,183 semantic alignment failures and 13,256 conditional proof failures.

To evaluate whether verification failures stem from downstream reasoning limits or latent upstream translation defects, PGD diagnoses all 13,256 submissions that pass semantic alignment but fail proof verification. Among these PGD cases, 12,488 (94.21%) are classified as proof errors, 273 (2.06%) as missing assumptions or conditions, and 495 (3.73%) as overly complex formalizations. Crucially, combining upfront semantic alignment rejections with the two formalization-related PGD categories, formalization failures account for 57.58% of all unsuccessful outputs. This fine-grained attribution confirms that while tactical search errors dominate once formalization is sound, formal translation defects remain the primary aggregate bottleneck across the full pipeline.

4.5 Inference-Time Verification and Revision

Beyond static evaluation, we investigate whether FormalMathArena can serve as an active verification harness to guide iterative refinement at test time. To demonstrate the effectiveness of our diagnostic feedback, we construct a comparative baseline using standard Lean compiler diagnostics on a held-out set of 400 problems across all ten models (Table 3). For each model, both settings initialize from identical initial generations, employ the same generator, and permit up to three revision rounds. Whereas the compiler baseline provides only raw type-checker error messages, FormalMathArena supplements these with semantic alignment feedback and proof-guided failure attributions. A problem is counted as successful if either its initial generation or any subsequent revision satisfies full end-to-end verification.

Across the 4,000 evaluated instances, FormalMathArena feedback drives the overall end-to-end success rate to 52.1%, achieving an average gain of 2.9 percentage points over the Lean compiler baseline at 49.2% and improving substantially upon the unrevised baseline of 39.1% (Table 3). In paired instance comparisons, FormalMathArena succeeds on 232 cases where compiler feedback fails to

Table 4: Pipeline outcomes and fine-grained PGD failure attributions across all submission. Percentages are calculated relative to each group's denominator. *Missing conditions* denotes missing assumptions or conditions.

Pipeline outcomes $N = 48{,}270$		
Outcome	Cases	%
SA failure	16,183	33.53
SA pass, proof failure	13,256	27.46
E2E success	**18,831**	**39.01**
PGD breakdown $N = 13{,}256$		
Attribution	Cases	%
Missing conditions	273	2.06
Overly complex	495	3.73
Proof error	12,488	94.21

produce a valid proof, compared to 117 cases in the opposite direction. This net advantage confirms that multi-dimensional semantic and causal feedback provides superior actionable guidance over raw syntax-level compiler errors.

At the individual model level, FormalMathArena delivers consistent improvements across models that lack complex internal scaffolds. The boost is especially pronounced for models with latent proving capacity, led by DeepSeek V4-Pro with an absolute gain of 12.2 percentage points over the Lean baseline and a 22.5 percentage point leap relative to its initial outputs. Solid gains are likewise observed for DeepSeek V4-Flash (+7.5 pp), Claude Opus 4.8 (+4.5 pp), and Kimi K2.7-Code (+3.5 pp). For general base models such as Qwen3.7-Max, FormalMathArena achieves performance comparable to the Lean baseline (-0.5 pp). The only notable degradation occurs in Qwen3-Coder 480B (-3.2 pp). This decline stems from architectural friction: Qwen3-Coder relies heavily on an intricate internal agent harness for formalization, and imposing FormalMathArena as an external verification harness creates coordination mismatches and prompt interference with its native scaffolding. Aside from this specific interaction conflict, our framework reliably enhances test-time proving performance across diverse model architectures.

5 Conclusion

We presented FormalMathArena, an end-to-end benchmark and diagnostic framework evaluating the full trajectory from natural language mathematics to verified Lean proofs. By integrating semantic alignment and proof-guided diagnosis, both validated against expert annotations at 91.50% and 86.00% accuracy, our pipeline uncovers failure modes obscured by binary compiler checks. Benchmarking across ten frontier LLMs reveals that formalization attrition constitutes the primary practical bottleneck, causing proof-only evaluations to overestimate real-world performance by up to 27.39 percentage points. Furthermore, deploying our diagnostic framework as an inference-time verification harness yields an overall gain of 2.9 percentage points over compiler-only feedback.

Limitations. Several limitations of this study warrant consideration. First, while our automated pipeline addresses the opacity of binary compiler checks, the semantic alignment (SA) and proof-guided diagnosis (PGD) modules inherently rely on LLM reasoning, where residual misclassifications may introduce noise into derived metrics. In addition, using GPT-5.5 as the diagnostic backbone introduces potential self-preference bias when evaluating outputs from the same model family, although consistent macro-level trends across evaluated architectures indicate that this factor does not alter our primary conclusions. Second, our framework is grounded in Lean 4 with a pinned Mathlib commit to ensure determinism and reproducibility, meaning our findings reflect specific formalization paradigms that may shift with library evolutions or vary in other proof assistants such as Isabelle/HOL or Coq. Finally, FormalMathArena centers primarily on competition-level and undergraduate mathematics, leaving open-ended research-level proving and long-horizon multi-step search for future investigation.

AI USE STATEMENT

We used the LLMs listed in the experimental setup to generate formal statements and proofs and to perform semantic alignment and proof-guided diagnosis. We also used AI tools to assist with manuscript drafting, language editing and translation, and figure preparation. The authors take responsibility for the final manuscript.

REPRODUCIBILITY STATEMENT

Experiments use Lean 4.21 with a fixed Mathlib environment. The supplementary material describes the dataset, generation and evaluation procedures, and expert validation, and includes the generation and SA prompts and detailed results. Code and data will be released upon acceptance.

REFERENCES

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. URL https://arxiv.org/abs/2302.12433.

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

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*, 2021. URL https://arxiv.org/abs/2110.14168.

Leonardo de Moura and Sebastian Ullrich. The Lean 4 theorem prover and programming language. In *Automated Deduction – CADE 28*, volume 12699 of *Lecture Notes in Computer Science*, pp. 625–635. Springer, 2021. doi: 10.1007/978-3-030-79876-5_37. URL https://doi.org/10.1007/978-3-030-79876-5_37.

Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. Measuring mathematical problem solving with the MATH dataset. In *Proceedings of the Neural Information Processing Systems Track on Datasets and Benchmarks*, volume 1, 2021. URL https://datasets-benchmarks-proceedings.neurips.cc/paper/2021/hash/be83ab3ecd0db773eb2dc1b0a17836a1-Abstract-round2.html.

Thomas Hubert, Rishi Mehta, Laurent Sartran, Miklós Z. Horváth, Goran Žužić, Eric Wieser, Aja Huang, Julian Schrittwieser, Yannick Schroecker, Hussain Masoom, Ottavia Bertolli, Tom Zahavy, Amol Mandhane, Jessica Yung, Iuliya Beloshapka, Borja Ibarz, Vivek Veeriah, Lei Yu, Oliver Nash, Paul Lezeau, Salvatore Mercuri, Calle Sönne, Bhavik Mehta, Alex Davies, Daniel Zheng, Fabian Pedregosa, Yin Li, Ingrid von Glehn, Mark Rowland, Samuel Albanie, Ameya Velingker, Simon Schmitt, Edward Lockhart, Edward Hughes, Henryk Michalewski, Nicolas Sonnerat, Demis Hassabis, Pushmeet Kohli, and David Silver. Olympiad-level formal mathematical reasoning with reinforcement learning. *Nature*, 651(8106):607–613, 2026. doi: 10.1038/s41586-025-09833-y. URL https://doi.org/10.1038/s41586-025-09833-y.

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

Jiedong Jiang, Wanyi He, Yuefeng Wang, Guoxiong Gao, Yongle Hu, Jingting Wang, Nailin Guan, Peihao Wu, Chunbo Dai, Liang Xiao, and Bin Dong. FATE: A formal benchmark series for frontier algebra of multiple difficulty levels. In *International Conference on Learning Representations*, 2026. URL https://arxiv.org/abs/2511.02872.

Daniel Kahneman. *Thinking, Fast and Slow*. Farrar, Straus and Giroux, 2011. ISBN 9780374275631. URL https://us.macmillan.com/books/9780374275631/thinkingfastandslow/.

Kyuhee Kim, Auguste Poiroux, and Antoine Bosselut. Do LLMs game formalization? evaluating faithfulness in logical reasoning. *arXiv preprint arXiv:2604.19459*, 2026. URL https://arxiv.org/abs/2604.19459. VerifAI-2 Workshop at ICLR 2026.

Lean Project. Lean Eval: Comparator-based Lean formal mathematics evaluation. https://github.com/leanprover/lean-eval, 2026. Accessed 2026-09-25.

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. *arXiv preprint arXiv:2502.07640*, 2025. URL https://arxiv.org/abs/2502.07640.

LLMTeamAkiyama. cleaned_NuminaMath-RL-Verifiable_with_proof. https://huggingface.co/datasets/LLMTeamAkiyama/cleaned_NuminaMath-RL-Verifiable_with_proof, 2025. Accessed 2026-09-25.

Xinghua Lou, Miguel Lázaro-Gredilla, Antoine Dedieu, Carter Wendelken, Wolfgang Lehrach, and Kevin P. Murphy. AutoHarness: Improving LLM agents by automatically synthesizing a code harness. *arXiv preprint arXiv:2603.03329*, 2026. URL https://arxiv.org/abs/2603.03329.

Jianqiao Lu, Yingjia Wan, Zhengying Liu, Yinya Huang, Jing Xiong, Chengwu Liu, Jianhao Shen, Hui Jin, Jipeng Zhang, Haiming Wang, Zhicheng Yang, Jing Tang, and Zhijiang Guo. Process-driven autoformalization in Lean 4. *arXiv preprint arXiv:2406.01940*, 2024. URL https://arxiv.org/abs/2406.01940. FormL4 dataset and code: https://github.com/rookie-joe/PDA.

Jianqiao Lu, Yingjia Wan, Yinya Huang, Jing Xiong, Zhengying Liu, and Zhijiang Guo. FormalAlign: Automated alignment evaluation for autoformalization. In *International Conference on Learning Representations*, 2025. URL https://proceedings.iclr.cc/paper_files/paper/2025/hash/fceedf8c9c0ff51f41b9fe0294ef0070-Abstract-Conference.html.

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. *arXiv preprint arXiv:2605.17283*, 2026. URL https://arxiv.org/abs/2605.17283.

Noor Islam S. Mohammad and Tamim Sheikh. The faithfulness gap: Certifying semantic equivalence between natural-language and formal mathematical statements. In *The 6th Muslims in ML (MusIML) Workshop at ICML 2026*, 2026. URL https://openreview.net/forum?id=bDKvdZJVgK.

Azim Ospanov, Farzan Farnia, and Roozbeh Mohit. miniF2F-Lean revisited: Reviewing limitations and charting a path forward. In *Advances in Neural Information Processing Systems*, volume 38, 2025. doi: 10.52202/085713-2671. URL https://proceedings.neurips.cc/paper_files/paper/2025/hash/72dad95a24fae750f8ab1cb3dab5e58d-Abstract-Conference.html.

Auguste Poiroux, Gail Weiss, Viktor Kunčak, and Antoine Bosselut. Reliable evaluation and benchmarks for statement autoformalization. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pp. 17947–17969, Suzhou, China, 2025. Association for Computational Linguistics. doi: 10.18653/v1/2025.emnlp-main.907. URL https://aclanthology.org/2025.emnlp-main.907/.

Zhizhou Ren, Zhihong Shao, Junxiao Song, Huajian Xin, et al. DeepSeek-Prover-V2: Advancing formal mathematical reasoning via reinforcement learning for subgoal decomposition. *arXiv preprint arXiv:2504.21801*, 2025. URL https://arxiv.org/abs/2504.21801.

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. In *Advances in Neural Information Processing Systems*, volume 37, pp. 11545–11569, 2024. doi: 10.52202/079017-0368. URL https://proceedings.neurips.cc/paper_files/paper/2024/hash/1582eaf9e0cf349e1e5a6ee453100aa1-Abstract-Datasets_and_Benchmarks_Track.html.

Haiming Wang, Mert Unsal, Xiaohan Lin, Mantas Baksys, Junqi Liu, Marco Dos Santos, Flood Sung, Marina Vinyes, et al. Kimina-Prover preview: Towards large formal reasoning models with reinforcement learning. *arXiv preprint arXiv:2504.11354*, 2025. URL https://arxiv.org/abs/2504.11354.

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, 2022. doi: 10.52202/068431-2344. URL https://proceedings.neurips.cc/paper_files/paper/2022/hash/d0c6bc641a56bebee9d985b937307367-Abstract-Conference.html.

Zijian Wu, Suozhi Huang, Zhejian Zhou, Huaiyuan Ying, Jiayu Wang, Dahua Lin, and Kai Chen. InternLM2.5-StepProver: Advancing automated theorem proving via expert iteration on large-scale LEAN problems. *arXiv preprint arXiv:2410.15700*, 2024. URL https://arxiv.org/abs/2410.15700v1.

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. *arXiv preprint arXiv:2405.14333*, 2024. URL https://arxiv.org/abs/2405.14333.

Kaiyu Yang, Aidan 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, 2023. URL https://proceedings.neurips.cc/paper_files/paper/2023/hash/4441469427094f8873d0fecb0c4e1cee-Abstract-Datasets_and_Benchmarks.html.

Huaiyuan Ying, Zijian Wu, Yihan Geng, Jiayu Wang, Dahua Lin, and Kai Chen. Lean Workbook: A large-scale Lean problem set formalized from natural language math problems. In *Advances in Neural Information Processing Systems*, volume 37, 2024. URL https://proceedings.neurips.cc/paper_files/paper/2024/hash/bf236666a2cc5f3ae05d2e08485efc4c-Abstract-Datasets_and_Benchmarks_Track.html.

Xuejun Yu, Jianyuan Zhong, Zijin Feng, Pengyi Zhai, Roozbeh Yousefzadeh, Wei Chong Ng, Haoxiong Liu, Ziyi Shou, Jing Xiong, Yudong Zhou, Claudia Beth Ong, Austen Jeremy Sugiarto, Yaoxi Zhang, Wai Ming Tai, Huan Cao, Dongcai Lu, Jiacheng Sun, Qiang Xu, Shen Xin, and Zhenguo Li. Mathesis: Towards formal theorem proving from natural languages. *arXiv preprint arXiv:2506.07047*, 2025a. URL https://arxiv.org/abs/2506.07047.

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*, 2025b. URL https://arxiv.org/abs/2505.02735.

Ke Zhang, Patricio Gallardo Candela, Sudhir Murthy, Yi Xie, Zhi Wang, and Maziar Raissi. Beyond compilation: Evaluating faithful natural-language-to-Lean statement formalization. *arXiv preprint arXiv:2606.31002*, 2026. URL https://arxiv.org/abs/2606.31002.

Kunhao Zheng, Jesse Michael Han, and Stanislas Polu. MiniF2F: A cross-system benchmark for formal olympiad-level mathematics. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=9ZPegFuFTFv.

Lianmin Zheng, Liangsheng Yin, Zhiqiang Xie, Chuyue Sun, Jeff Huang, Cody Hao Yu, Shiyi Cao, Christos Kozyrakis, Ion Stoica, Joseph E. Gonzalez, Clark Barrett, and Ying Sheng. SGLang: Efficient execution of structured language model programs. In *Advances in Neural Information Processing Systems*, volume 37, pp. 62557–62583. Curran Associates, Inc., 2024. doi: 10.52202/079017-2000. URL https://proceedings.neurips.cc/paper_files/paper/2024/file/724be4472168f31ba1c9ac630f15dec8-Paper-Conference.pdf.

A Appendix

The supplementary material provides complete metric derivations, dataset and generation settings, SA acceptance checks, expert-validation confusion matrices, detailed model results, PGD attributions, paired revision statistics, and the generation and SA prompts.