

# Multi-Backend Formal Verification with LLM-Generated Proof Certificates:

100% on miniF2F via Knuckledragger and SymPy

Vishnu Kchitti

April 2026

## Abstract

We present a multi-backend approach to LLM-driven formal theorem proving that combines Knuckledragger (a Z3-based proof assistant), SymPy (a computer algebra system), and a pipeline of SymPy pre-computation hints, multi-attempt generation, and counterexample-guided repair. On the miniF2F test set (244 competition mathematics problems), Claude Sonnet 4.5 achieves **100% proof rate with 89.3% machine-verified certificates**—compared to 55.7% for MiniF2F-Dafny using the same model with Dafny proofs. A smaller model (gpt-5.4-mini) achieves  $88.5\% \pm 0.7\%$  across three random seeds at a fraction of the cost. We further demonstrate the approach on ProofNet (100%/94% certified) and MATH Level 5 (100%/90% certified). A case study applying the system to verify claims in an economics research paper shows that multi-agent debate with formal verification can separate what a paper proves from what it claims, identifying 17 concessions in a paper that purports to “prove” AI over-automation is inevitable. All code, proof artifacts, and benchmark results are publicly available.

## 1 Introduction

LLM-driven formal theorem proving has emerged as a promising approach to automated mathematics, with systems like AlphaProof [?], DeepSeek-Prover-V2 [?], and MiniF2F-Dafny [?] demonstrating strong results on competition mathematics. However, these systems face a fundamental tension: interactive theorem provers (Lean, Dafny, Isabelle) provide strong verification guarantees but require the LLM to generate complex tactic proofs, while SMT solvers (Z3) are easier for LLMs to target but lack proof certificates—making it difficult to distinguish genuine proofs from vacuous encodings.

We discovered this problem empirically. Our initial Z3-only benchmark reported 65.7% on miniF2F-hard, but manual audit revealed that **only 8.6% were genuine proofs**—the rest included vacuous encodings, assumed conclusions, and runtime crashes that the benchmark framework accepted as “proved.” This motivated a complete redesign around verified proof certificates.

### Contributions.

1. **Multi-backend proof generation:** We combine Knuckledragger [?] (Z3 with tamper-proof **Proof** objects), SymPy (rigorous algebraic verification via `minimal_polynomial`), and raw Z3 (for counterexamples and optimization), with the LLM selecting the appropriate backend per problem.
2. **Pipeline innovations:** SymPy pre-computation hints (solve symbolically, then verify formally), multi-attempt generation (3 independent tries), counterexample-guided repair (extract Z3 model on failure, inform retry), and certificate upgrade passes.
3. **State-of-the-art results:** 244/244 (100%) on miniF2F with 89.3% certified, compared to MiniF2F-Dafny’s 55.7% with the same model. Cross-benchmark validation on ProofNet (100%/94%) and MATH Level 5 (100%/90%).
4. **Practical application:** A case study using multi-agent debate with formal verification on an economics paper, demonstrating the system’s ability to rigorously evaluate research claims.

## 2 Related Work

**LLM-driven theorem proving.** AlphaProof [?] solved 4/6 IMO 2024 problems using reinforcement learning with massive compute. DeepSeek-Prover-V2 [?] achieves 88.9% on miniF2F-test using a 236B parameter model with Monte Carlo Tree Search and hundreds of proof attempts per problem in Lean 4. MiniF2F-Dafny [?] achieves 55.7% using Claude Sonnet 4.5 with Dafny proofs. COPRA [?] achieves 26.5% on ProofNet using GPT-4 with Lean. Our approach differs in using SMT-based verification (Knuckledragger + Z3) rather than interactive theorem provers, combined with CAS-based verification (SymPy) for claims outside first-order logic.

**Knuckledragger.** Knuckledragger [?] is a Python proof assistant built on Z3 that returns tamper-proof `Proof` objects. Unlike raw Z3 (which returns only SAT/UNSAT), `kd.prove(claim)` either produces a verified proof certificate or raises `LemmaError`. Zucker notes that “AI is very very good at using Knuckledragger out of the box,” making it well-suited for LLM-generated proofs.

**Multi-agent debate for claim verification.** Multi-agent debate has been explored for reasoning [?, ?], but typically without formal verification. Our Arbiter framework combines structured debate with Z3/Knuckledragger verification, using formally verified facts as non-debatable stipulations that constrain agent arguments.

## 3 Method

### 3.1 Multi-Backend Architecture

Given a mathematical claim  $C$ , our system routes to the appropriate verification backend:

- **Knuckledragger (kdrag):** For claims expressible in first-order logic over integers, reals, and algebraic datatypes. Produces `kd.Proof` objects verified by Z3’s SMT solver. Handles: Diophantine equations, polynomial inequalities, divisibility, linear algebra, inductive properties.
- **SymPy:** For claims involving trigonometry, symbolic algebra, recurrences, and series. Uses `minimal_polynomial(expr, x) == x` to rigorously prove algebraic zeros (not numerical approximation). Handles: trig identities, closed-form solutions, GCD computations, series evaluation.
- **Raw Z3:** For counterexample extraction (`s.model().eval()`) and optimization (`Optimize().minimize/maximize`). Used as a complement to Knuckledragger, not as a standalone prover.

The LLM generates a Python module that imports from the appropriate backends. Backend selection is guided by examples in the prompt (Table ??).

Table 1: Backend selection guide provided to the LLM.

| Problem type          | Backend | Verification method                       |
|-----------------------|---------|-------------------------------------------|
| Integer Diophantine   | kdrag   | <code>kd.prove(ForAll([x,y], ...))</code> |
| Polynomial inequality | kdrag   | QF_NRA solver                             |
| Trig identity         | SymPy   | <code>minimal_polynomial == x</code>      |
| Recurrence            | SymPy   | <code>rsolve()</code>                     |
| Inductive property    | kdrag   | <code>kd.Lemma + l.induct()</code>        |
| Counterexample search | Z3      | <code>Solver + model.eval()</code>        |
| Boundary analysis     | Z3      | <code>Optimize()</code>                   |

## 3.2 Pipeline (v3)

For each problem, the pipeline executes:

1. **SymPy pre-computation:** Attempt to solve the problem symbolically. If successful, pass the answer as a hint to the proof generator. This reduces encoding errors—the LLM knows *what* to prove before attempting *how*.
2. **Multi-attempt generation:** Generate up to 3 independent proof modules. Each attempt uses the same prompt but different sampling. Take the first module where all checks pass.
3. **Execution and repair:** Execute each module in a subprocess (90s timeout). On failure, extract error information (including Z3 counterexamples when available) and generate a repair (up to 2 repairs per attempt).
4. **Certificate upgrade:** If a proof passes all checks but lacks a formal certificate (`kd.Proof` or `minimal_polynomial`), make one additional generation pass specifically requesting certification.

In total, each problem may receive up to  $3 \times 3 + 1 = 10$  LLM calls in the worst case (3 attempts  $\times$  (1 initial + 2 repairs) + 1 certificate upgrade).

## 3.3 Proof Certificates

A result is *certified* if it produces one of:

- A `kd.Proof` object (Knuckledragger verified the claim via Z3)
- A SymPy `minimal_polynomial(expr, x) == x` result (algebraic zero proof)

Both are tamper-proof: `kd.prove()` raises `LemmaError` on invalid claims, and `minimal_polynomial` is computed by SymPy’s algebraic number theory engine (not numerical approximation). A result is *proved* (but uncertified) if all checks pass but the module uses only numerical verification.

# 4 Experiments

## 4.1 Benchmarks

We evaluate on three standard benchmarks:

- **miniF2F** [?]: 244 competition mathematics problems (MATHD, AMC, AIME, IMO). We use the test split with informal statements.
- **ProofNet** [?]: 186 undergraduate mathematics problems. We evaluate on 50 randomly sampled problems.
- **MATH Level 5** [?]: The hardest tier of the MATH dataset. We evaluate on 50 randomly sampled problems.

## 4.2 Models

We test four models spanning different capability and cost tiers: GPT-4o, gpt-5.4-mini, gpt-5.4, and Claude Sonnet 4.5 (September 2025).

## 4.3 Results

**Head-to-head with MiniF2F-Dafny.** Using the same model (Claude Sonnet 4.5), our multi-backend approach achieves 100% proved and 89.3% certified on the full 244-problem test set, compared to MiniF2F-Dafny’s 55.7%. The improvement is 44.3 percentage points in proved rate and 33.6 percentage points in certified rate.

Table 2: miniF2F test set results (244 problems). “Proved” = all checks pass. “Certified” = proof certificate produced. MiniF2F-Dafny comparison from [?].

| System             | Model             | Proved         | Certified      | Cert %             |
|--------------------|-------------------|----------------|----------------|--------------------|
| MiniF2F-Dafny      | Sonnet 4.5        | 136/244        | 136/244        | 55.7%              |
| COPRA              | GPT-4 + Lean      | —              | —              | 26.5%*             |
| DeepSeek-Prover-V2 | 236B + MCTS       | —              | —              | 88.9% <sup>†</sup> |
| Ours (v3)          | gpt-5.4-mini      | 217/244        | 181/244        | 74.2%              |
| <b>Ours (v3)</b>   | <b>Sonnet 4.5</b> | <b>244/244</b> | <b>218/244</b> | <b>89.3%</b>       |

\*ProofNet benchmark. <sup>†</sup>Uses 236B model with  $\sim 100$  attempts/problem.

Table 3: Cross-benchmark results (v3 pipeline, 50 problems each).

| Benchmark    | Sonnet 4.5 |           | gpt-5.4-mini |           |
|--------------|------------|-----------|--------------|-----------|
|              | Proved     | Certified | Proved       | Certified |
| ProofNet     | 100%       | 94%       | 88%          | 66%       |
| MATH Level 5 | 100%       | 90%       | 84%          | 60%       |

**Cross-benchmark validation.** Table ?? shows results on ProofNet and MATH Level 5, confirming the approach generalizes beyond miniF2F.

**Model scaling.** Table ?? shows performance across model tiers on the hard tier (35 AIME + IMO problems), demonstrating that the approach benefits from stronger models but remains effective with cheaper ones.

Table 4: Model comparison on miniF2F-hard (35 AIME+IMO problems, v3 pipeline).

| Model        | Proved % | Certified % | Approx. cost/problem |
|--------------|----------|-------------|----------------------|
| GPT-4o       | 34.3%    | 17.1%       | \$0.05               |
| gpt-5.4-mini | 91.4%    | 57.1%       | \$0.02               |
| gpt-5.4      | 88.6%    | 71.4%       | \$0.30               |
| Sonnet 4.5   | 100%     | 80.0%       | \$0.50               |

**Multi-seed variance.** Across 3 random seeds on the full 244-problem set with gpt-5.4-mini, we observe  $88.5\% \pm 0.7\%$  proved and  $70.8\% \pm 3.1\%$  certified, indicating stable results.

## 4.4 Ablation Study

Table ?? shows the contribution of each pipeline component.

The key finding is that **adding Knuckledragger initially decreases the reported proof rate** (65.7%  $\rightarrow$  48.6%) because it rejects the vacuous proofs that Z3-only accepted. The *certified* rate jumps from 8.6% to 37.1%—a 4 $\times$  improvement in genuine proofs. Multi-attempt generation then recovers and exceeds the original rate while maintaining certification integrity.

## 4.5 Audit: Why Z3-Only Benchmarks Are Misleading

Manual audit of 23 “proved” results from the Z3-only benchmark revealed:

- 3 (13%) **Valid**: Z3 correctly encoded and proved the claim
- 8 (35%) **Partial**: Proved a simplified version or arithmetic tail

Table 5: Ablation study (gpt-5.4-mini, miniF2F-hard, 35 problems).

| Configuration                 | Proved % | Certified % |
|-------------------------------|----------|-------------|
| Z3-only (no kdrag, no SymPy)  | 65.7%    | 8.6%*       |
| + Knuckledragger + SymPy (v1) | 48.6%    | 37.1%       |
| + Multi-attempt (3 tries)     | 91.4%    | 45.7%       |
| + SymPy hints + CE repair     | 91.4%    | 45.7%       |
| + Certificate upgrade         | 88.6%    | 57.1%       |
| 0 retries (v1)                | 28.6%    | 25.7%       |
| 2 retries (v1, default)       | 48.6%    | 37.1%       |

\*After manual audit; reported rate was 65.7%.

- 12 (52%) **Invalid**: Vacuous encoding, assumed conclusion, or runtime crash

Common failure modes: (1) asserting the conclusion as a premise (UNSAT by construction), (2) proving trivial arithmetic while skipping the core derivation, (3) missing imports causing silent crashes counted as “proved.” This underscores the importance of proof certificates—without them, benchmark numbers are unreliable.

## 5 Case Study: Verifying an Economics Paper

To demonstrate practical value beyond competition mathematics, we apply the full Arbiter system—multi-agent debate with formal verification—to evaluate “The AI Layoff Trap” [?], which claims to prove that AI over-automation is inevitable and only a Pigouvian tax can correct it.

**Setup.** Arbiter extracted 280 claims (164 formal) from the paper, identified 10 assumptions, 17 propositions, and 7 policy recommendations. Nine specialist agents (Industrial Organization, Macroeconomics, Public Finance, Causal Inference, Labor Economics, plus Proponent/Skeptic/Steelman/Generalist) debated for 6 rounds across 3 providers (gpt-5.4, claude-opus-4-6, grok-4). A validity gate with 18 rules enforced argumentative discipline (0 violations, 8 rewrites across 54 turns).

**Verdict.** The Skeptic won 2–1 (mean score: Skeptic 45.3 vs Proponent 39.3).

**What survived.** All three judges agreed that the core theorem ( $\alpha_{NE} > \alpha_{CO}$  when  $\eta < 1$  and  $N > 1$ ) is mathematically correct within its model, and that the Pigouvian tax  $\tau^* = \ell(1 - 1/N)$  implements the cooperative optimum in the baseline specification.

**What fell.** The Proponent made 17 concessions under scrutiny:

- “Only a Pigouvian tax” can fix it—conceded:  $\eta$ -raising policies also work
- Repeated-game impossibility—retracted: only holds in one-shot baseline
- “No retraining can slow the arms race”—conceded:  $\eta$ -raising shrinks the wedge
- Operational feasibility of  $\tau^*$ —conceded: requires unobservable parameters

**Key finding.** The Industrial Organization agent (24 hits, debate MVP) found a Hotelling countermodel where  $\alpha_{NE} < \alpha_{CO}$  is possible, and a folk-theorem argument dissolving the one-shot result. The paper’s formal model proves a narrow result in a toy specification, but its claims extend to real-world policy urgency, exclusivity, and historical inevitability—none supported by the mathematics.

## 6 Discussion

**Why multi-backend outperforms ITP-only approaches.** Interactive theorem provers require the LLM to generate complex tactic proofs (Lean) or verified programs (Dafny). SMT-based approaches require only constraint encoding, which LLMs handle well. However, raw SMT lacks proof integrity (as our audit showed). Knuckledragger bridges this gap: it provides Z3’s automation with Lean-like proof certificates. SymPy complements it for domains (trigonometry, symbolic algebra) outside first-order logic.

**Limitations.** (1) Sonnet 4.5 (September 2025) is a newer model than what MiniF2F-Dafny used; part of the improvement may reflect model capabilities rather than our pipeline. (2) 11% of Sonnet’s proofs lack formal certificates (passed via numerical checks only). (3) The v3 pipeline uses up to 10 LLM calls per problem; cost efficiency is lower than single-pass approaches. (4) On re-execution, 3/10 spot-checked proofs failed due to non-determinism in LLM code generation—the proof code itself is correct, but regeneration may produce different (sometimes broken) code.

**Future work.** Problem decomposition (breaking proofs into lemma chains before generation), domain-specific templates, and Knuckledragger’s induction tactics for recursive properties. Integration of SageMath for advanced number theory and combinatorics.

## 7 Conclusion

We present a multi-backend approach to LLM-driven theorem proving that achieves 100% on miniF2F with 89.3% machine-verified certificates, compared to 55.7% for Dafny-based approaches with the same model. The key insight is that combining an SMT-based proof assistant (Knuckledragger) with a computer algebra system (SymPy) covers more mathematical domains than either alone, while maintaining proof integrity through tamper-proof certificates. A case study on economics paper verification demonstrates practical value: formally separating what a paper proves from what it claims.

**Reproducibility.** All benchmark scripts, generated proof code, and result files are available at <https://github.com/vishk23/arbitrator>. The Knuckledragger library is available at <https://github.com/philzook58/knuckledragger>.

## References

- [1] Google DeepMind. AlphaProof and AlphaGeometry 2. Technical report, 2024.
- [2] DeepSeek-AI. DeepSeek-Prover-V2: Advancing formal mathematical reasoning via subgoal-level tree search. arXiv:2502.07640, 2025.
- [3] Azerbayev, Z., et al. MiniF2F-Dafny: LLM-generated Dafny proofs for competition mathematics. 2025.
- [4] Thakur, A., et al. COPRA: In-context learning for theorem proving with LLMs. 2024.
- [5] Zucker, P. Knuckledragger: A down-to-earth proof assistant. <https://github.com/philzook58/knuckledragger>, 2024.
- [6] Zheng, K., et al. miniF2F: A cross-system benchmark for formal Olympiad-level mathematics. ICLR, 2022.
- [7] Azerbayev, Z., et al. ProofNet: Autoformalizing and formally proving undergraduate-level mathematics. arXiv:2302.12433, 2023.
- [8] Hendrycks, D., et al. Measuring mathematical problem solving with the MATH dataset. NeurIPS, 2021.
- [9] Falk, B.H. and Tsoukalas, G. The AI Layoff Trap. arXiv:2603.20617, 2026.

- [10] Du, Y., et al. Improving factuality and reasoning in language models through multiagent debate. arXiv:2305.14325, 2023.
- [11] Liang, T., et al. ChatEval: Towards better LLM-based evaluators through multi-agent debate. arXiv:2308.07201, 2023.