VERITAS: Verifier-Guided Proof Search for Zero-Shot Formal Theorem Proving

Manish Acharya [1] Zhenyu Liao [2] Yueke Zhang [1] Kevin Leach [1] Yu Huang [1] Yifan Zhang [1]

[1]Department of Computer Science, Vanderbilt University [2]Amazon, Seattle

Motivation

Formal theorem proving produces proofs that a computer can check rigorously. Large language models can propose proof steps, but their suggestions often contain invalid tactics, incorrect lemma names, or incomplete reasoning.

Lean provides detailed feedback about these failures. However, many proof-search pipelines reduce this information to a simple success-or-failure outcome.

Can we use errors and partial proof progress to guide the next attempt?

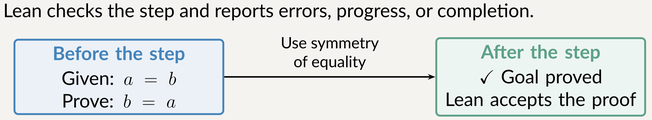
Key Idea: Turn Failed Attempts into Guidance

VERITAS uses Lean's errors and partial proof progress to guide the next attempt.

It first tries direct candidates, then searches unsolved theorems using the feedback collected.

How Lean Checks a Proof

A **proof state** records what is known and what remains to be proved. A **tactic** changes this state; Lean checks the step and reports errors, progress, or completion.



When a step fails: Given $h : a = b$, trying `exact h` for the goal $b = a$ produces a type mismatch. Using symmetry, `exact h.symm`, closes the goal.

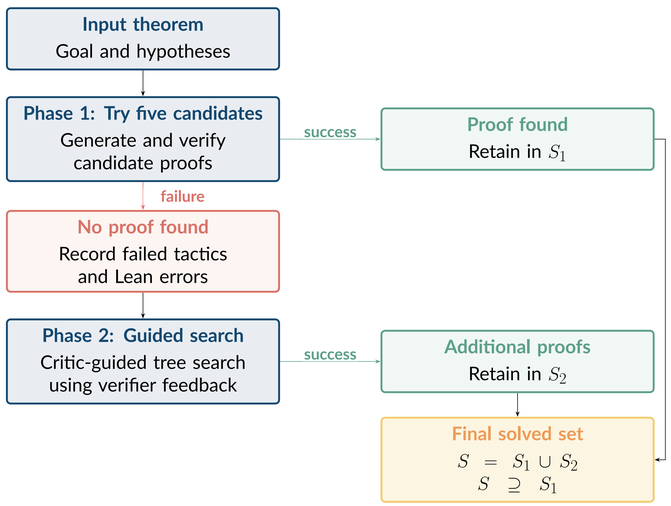


Figure 1. VERITAS two-phase search. Phase 1 retains successful proofs. Unsolved theorems proceed to verifier-guided search using failed tactics and Lean errors as context.

Method

VERITAS coordinates four specialized roles through a shared proof state. Each search expansion uses the current goal, relevant lemmas, and feedback from previous attempts. The guided search runs only on unsolved theorems, preserving every proof found during the initial sampling stage.

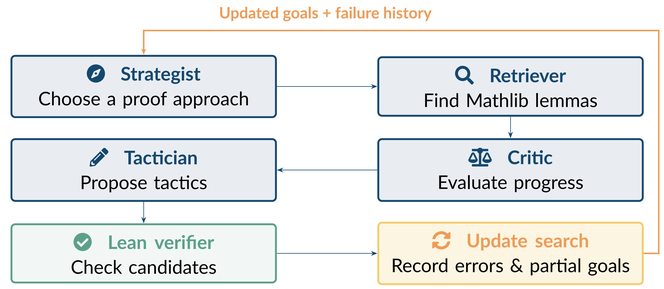


Figure 2. Verifier-guided generation. Lean's errors and partial goals guide the next search expansion.

How Search Chooses the Next Step

A search-tree node represents a partial proof state; an edge represents a proposed tactic.

Critic-guided Monte Carlo Tree Search balances:

- **Observed value:** how useful a branch has been.
- **Exploration:** whether a branch has received few visits.
- **Critic value:** how promising the resulting state appears.
- **Strategy alignment:** whether the tactic matches the selected approach.

$$\mathrm{Score}(s,a) = Q(s,a) + c\sqrt{\frac{\ln N(s)}{N(s,a)}} + w_v V_{\mathrm{critic}}(s') + w_s\ \mathrm{align}(a,\pi).$$

Here s' is the state reached by tactic a; N records visits, and π is the chosen strategy.

Lean's syntax, type, and progress signals also contribute to state evaluation, preserving useful intermediate progress.

Efficient Verification

At each expansion, VERITAS places the six candidate checks in one Lean file and verifies them in a single invocation.

$$\underbrace{K \text{ separate Lean calls}}_{\text{individual checks}} \longrightarrow \underbrace{1 \text{ Lean call}}_{\text{batched checks}}$$

Batched checking reduces repeated compiler-startup overhead, with approximately $10\times$ speedup reported at $K = 6$.

Search configuration: five initial candidates, followed by up to 50 guided-search iterations with six candidates per expansion.

Experiments & Results

We test on miniF2F and CombiBench against Portfolio and Best-of-1/5 baselines.

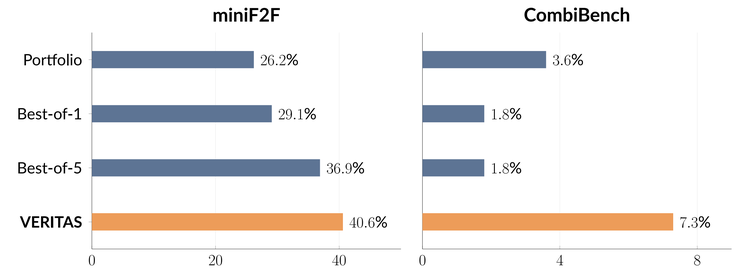


Figure 3. Benchmark solve rates. VERITAS reaches 40.6% on miniF2F and 7.3% on CombiBench. Both panels use the same method order but different horizontal scales.

What Guided Search Adds

Outcome	Theorems
Both methods	47
VERITAS only	**11**
Independent Best-of-5 only	5
Neither method	138
Total: hard miniF2F	**201**

11 VERITAS-only proofs remained unsolved by Best-of-25.

Conclusion & Future Work

- Lean's errors and partial goals inform subsequent proof attempts.
- Guided exploration finds additional multi-step proofs under the tested budgets.
- VERITAS uses structured feedback at inference time.

Future Work:

- Improve lemma retrieval, strengthen proof-state evaluation, and reduce inference cost.

Author Contributions & Acknowledgments

Manish Acharya proposed the research direction, and Yifan Zhang suggested the algorithmic approach. They jointly developed the algorithm. Manish implemented the framework, conducted the experiments, and drafted the manuscript; Yifan contributed to its final preparation. This research was supported by the SyBBURE Searle Undergraduate Research Fellowship.

References

[1] L. de Moura and S. Ullrich. *The Lean 4 Theorem Prover and Programming Language.* CADE, 2021.

[2] K. Zheng, J. M. Han, and S. Polu. *miniF2F: A Cross-System Benchmark for Formal Olympiad-Level Mathematics.* ICLR, 2022.