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

Anonymous ACL submission

Abstract

LLM-based formal provers collapse a verifier's rich signals (syntax errors, type mismatches, partial goal progress) into a binary pass/fail bit, discarding most of what the verifier actually says. We present **VERITAS**, a zero-shot framework that routes every verifier signal back into proof search through a two-phase protocol: Best-of-N sampling first, then a critic-guided MCTS pass that ingests Phase 1 failures as explicit negative examples. The protocol preserves every theorem solved by its own Phase 1 sweep, so Phase 2's additional solves are attributable to feedback-driven exploration. VERITAS reaches **40.6%** on miniF2F (vs. an independently run Best-of-5 at 36.9%, Portfolio 26.2%) and **7.3%** on **VERITAS-CombiBench**, a 55-theorem combinatorics benchmark we release on which Best-of-5 (1.8%) falls *below* Portfolio (3.6%), exposing that unguided sampling hurts when correct lemma names must be recovered iteratively from verifier feedback.

1 Introduction

Formal theorem proving, the construction of machine-checkable proofs in interactive proof assistants such as Lean, Coq, and Isabelle, is a formal-language generation problem: a model must emit tactic programs in a precise syntax whose semantics are checked by a deterministic verifier. It has become a flagship benchmark for evaluating language models on multi-step formal thought because success admits a crisp definition: a candidate proof is either accepted by the verifier or it is not. This property has driven a steady stream of progress, pushing solve rates on miniF2F (Zheng et al., 2022) from below 30% to above 80% (Polu and Sutskever, 2020; Yang et al., 2023; Wu et al., 2024b; Ren et al., 2025). Two complementary families drive this progress: language models that *generate* candidate tactics (Polu and Sutskever, 2020; Han et al., 2022; Yang et al., 2023; Azerbayev et al., 2024), and tree-search methods that *explore* the combinatorial space of partial proofs (Kocsis and Szepesvári, 2006; Silver et al., 2018; Lample et al., 2022; Google DeepMind, 2024). Most modern systems combine the two: an LM proposes, search disposes.

What is striking about this combination is how thinly the two components *communicate*. In a typical pipeline, the LM emits a batch of candidate tactics, the verifier checks each one, and the search algorithm updates value estimates from the resulting pass-or-fail outcomes. Anything richer that the verifier produces along the way (a parse error pointing at a malformed token, a type mismatch naming the offending hypothesis, a partial goal that has been simplified but not closed) is summarised into a scalar reward or thrown away entirely. Yet these intermediate signals are exactly the artefacts a human prover attends to: collapsing them into a single bit forces the search to relearn what the verifier has already told it explicitly. The cost is most visible on proofs that require a chain of *dependent* tactics, where each step's correctness hinges on the previous step's effect on the goal. Recent rich-feedback methods in RL (Hübotter et al., 2026) make the same observation in the post-training regime; we explore the dual at inference time.

We present **VERITAS**, a zero-shot framework built around a single design commitment: *the verifier's structured feedback should reach every generation decision*. We factor proof search into four specialised agents sharing a unified proof state. A *Strategist* chooses a high-level approach (direct computation, induction, rewriting, contradiction) before any tactics are emitted, narrowing the search to a strategy-relevant subspace. A *Tactician* (Claude Sonnet) proposes tactic candidates conditioned on the current goal, retrieved Mathlib premises, and crucially the explicit list of tactics that have already failed earlier in the search, with

their Lean error messages attached as negative examples. A *Critic* (Claude Haiku) scores partial proof states, replacing MCTS's random rollouts with a value estimate that integrates Lean's four-way feedback signal. A *Retriever* grounds generation in Mathlib by surfacing relevant lemmas. These agents are coordinated by a deliberately conservative two-phase protocol: Phase 1 is identical to a Best-of-N sweep, and Critic-guided MCTS in Phase 2 runs only on theorems Phase 1 failed. This yields a *monotonicity guarantee*: VERITAS preserves every theorem solved by its own Phase 1 attempts. The reported Best-of-5 baseline is an independent stochastic run, so its overlap with VERITAS need not be a subset; within a VERITAS run, however, every Phase 2-only solve is attributable to MCTS with verifier feedback. A batched Lean validation trick further cuts per-expansion verification from $O(K)$ to $O(1)$ proof-checker invocations.

Across two Lean 4 benchmarks the experiments give a consistent picture: tightly coupling the verifier with generation matters more than scaling either component alone. On miniF2F, VERITAS reaches 40.6% (99/244), above Best-of-5 Claude (36.9%) and a handcrafted Portfolio (26.2%); a phase-wise decomposition identifies 11 theorems not reached by our tested flat-sampling baselines. The picture sharpens on **VERITAS-CombiBench**: there, Best-of-5 (1.8%) falls *below* the Portfolio (3.6%) because unguided sampling hallucinates Mathlib lemma names Lean rejects on sight, while VERITAS reaches 7.3% by correcting those names iteratively from Lean's errors. We read these results as evidence for a general design principle: when a deterministic verifier emits structured intermediate signals, folding them back into generation amplifies the LM substantially more than additional sampling does.

Contributions. (i) A zero-shot four-agent framework that routes Lean's structured feedback (syntax/type/progress/completion) into every generation decision; (ii) a two-phase Best-of-N $\rightarrow$ Critic-MCTS protocol with an internal monotonicity guarantee that isolates the search contribution; (iii) a batched Lean validation trick that reduces verification cost by $10\times$; (iv) **VERITAS-CombiBench**, a benchmark of 55 hard, author-verified Lean 4 combinatorics theorems we release; and (v) a phase-wise attribution showing 11 miniF2F theorems uniquely solvable by MCTS, plus a failure-mode taxonomy that locates the next frontier for verifier-in-the-loop search.

2 The VERITAS Framework

VERITAS coordinates four specialized agents (Strategist, Tactician, Critic, and Retriever) through a shared proof state and a Critic-guided MCTS loop, with all agents observing the same structured verification signals from Lean. Figure 1 shows the full pipeline and Algorithm 1 gives the protocol in pseudo-code.

Proof state and verifier signals. All agents operate on a shared `ProofState`

$$s = (\text{theorem},\ \text{goal},\ H,\ \tau_{1:t},\ \sigma_A, \sigma_B, \sigma_C, \sigma_D),$$

where H is the hypothesis set, $\tau_{1:t}$ is the tactic history, and the four σ values are Lean's structured signals: σ_A syntax validity, σ_B type correctness, σ_C goal progress (e.g. subgoal reduction), and σ_D proof completion (no remaining goals or `sorry`). Every agent's output conditions the next agent's input within an iteration.

Agents. The *Strategist* maps a proof state to one of six high-level strategies (`direct`, `induction`, `contradiction`, `cases`, `rewrite`, `apply_lemma`) together with a depth estimate, restricting subsequent search to a strategy-relevant subspace. The *Tactician* (Claude Sonnet 4.6, temperature 0.8) generates K tactic candidates conditioned on the strategy, retrieved premises, Critic guidance, and crucially the failure set $\mathcal{F}$ of tactics rejected earlier (injected into the system prompt as negative examples). The *Critic* (Claude Haiku 4.5, temperature 0.3) returns a value $V(s) \in [0, 1]$, suggested next tactics, and a pruning recommendation, providing low-latency guidance for selection. The *Retriever* surfaces the top k=5 Mathlib premises by weighted keyword overlap over goal tokens, lemma names, and statement signatures (matching type names such as `Nat`/`Finset`, operators $+, *, \leq$, and connectives $\wedge, \vee$); results are appended to the Tactician's prompt and passed to the Critic as context.

2.1 Critic-Guided MCTS

Selection. We extend standard UCB with Critic value and strategy alignment, scoring an action a at state s as $Q(s, a) + c\sqrt{\ln N(s)/N(s, a)} + w_v V_{\text{critic}}(s') + w_s\,\text{align}(a, \pi)$, where $\text{align}(a, \pi) \in$

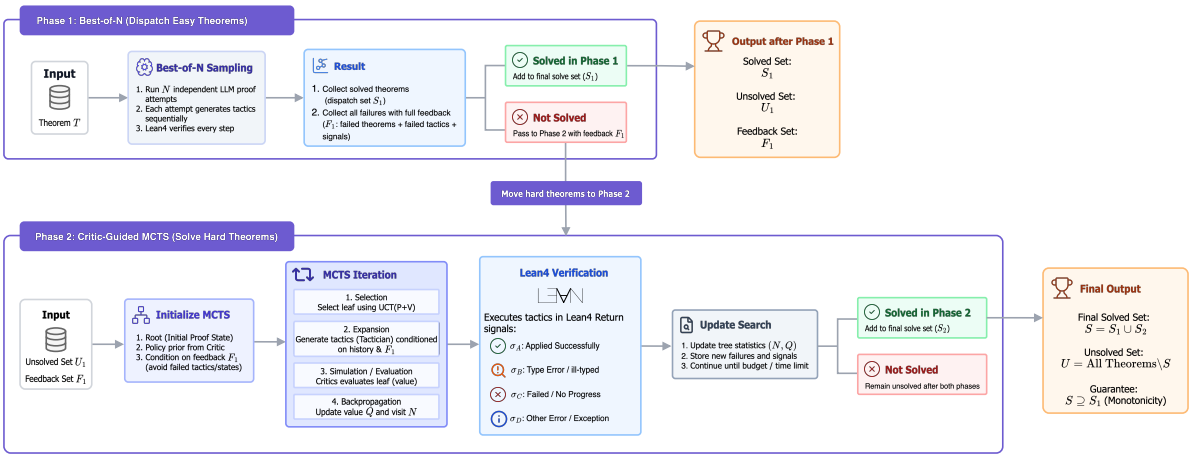

Figure 1: **VERITAS two-phase search protocol.** *Phase 1* (top): N independent LM proof attempts are verified by Lean 4. Solved theorems form the dispatch set S_1; failures (with full tactic histories and Lean signals) are collected as feedback corpus $\mathcal{F}_1$. *Phase 2* (bottom): unsolved theorems in U_1 enter Critic-guided MCTS with Selection (UCB with value $Q + V$), Expansion (Tactician conditioned on $\mathcal{F}_1$), Simulation (Critic value), Backpropagation. Lean 4 returns one of four signals: σ_A (syntax-valid application), σ_B (type-correct state), σ_C (goal progress), and σ_D (proof complete). The final solve set $S = S_1 \cup S_2$ satisfies the monotonicity guarantee $S \supseteq S_1$.

$\{0, 1\}$ equals 1 iff the leading token of a belongs to the strategy-specific tactic vocabulary $\mathcal{V}_\pi$ (e.g., $\mathcal{V}_{\text{induction}} = \{\texttt{induction}, \texttt{cases}, \texttt{rcases}\}$), biasing selection toward strategy-aligned subtrees. *Expansion* runs the four agents in sequence (Strategist $\rightarrow$ Retriever $\rightarrow$ Critic $\rightarrow$ Tactician), validates all K generated candidates in a single batched Lean call (Section 2.3), and creates child nodes for the syntactically valid ones. *Simulation* replaces random rollouts with the Critic's value augmented by the structured signals, $V(s) = V_{\text{critic}}(s) + \alpha_1\sigma_A + \alpha_2\sigma_B + \alpha_3\sigma_C + \beta/(1+N(s))$, with $\alpha_1{=}\alpha_2{=}0.1$ and $\alpha_3{=}0.2$ assigning increasing weight to increasingly informative signals (goal progress is the strongest intermediate signal short of completion) and $\beta{=}0.1$ providing an exploration bonus that decays with visit count. *Backpropagation* carries the structured A–B–C–D signal upward rather than collapsing to binary win/loss; this preserves credit for partial-progress states (σ_C) that would otherwise be pruned: the critical case for multi-step theorems.

2.2 Two-Phase Design and Monotonicity Guarantee

Phase 1 (Best-of-N sweep) generates N tactics in one Claude API call and validates them in one batched Lean call; if any succeeds, return immediately. **Phase 2 (Critic-guided MCTS)** activates only if Phase 1 fails, with the Phase 1 failure set $\mathcal{F}$ injected into the Tactician's system prompt.

Proposition 2.1 (Monotonicity)**.** *For any fixed VERITAS run on problem set $\mathcal{T}$ and Phase 1 budget N, the final VERITAS solve set is a superset of that run's own Phase 1 solve set: $S(\mathcal{T}) \supseteq S_1(\mathcal{T})$.*

Proof. If VERITAS's Phase 1 solves a theorem, that proof is immediately kept and Phase 2 is never invoked for that theorem. If Phase 1 fails, Phase 2 may or may not succeed, but it cannot remove a theorem already in S_1. □

This guarantee is an internal attribution property: every theorem in S_2 is a Phase 2 contribution. It does not imply that a separately sampled Best-of-5 baseline must be a subset of VERITAS, because the baseline is an independent stochastic run.

2.3 Batched Lean Validation

Rather than invoking Lean separately for each of K candidates, we compile them into one `.lean` file as private theorems with distinct names (`private theorem _p0 ... := by <tactic_0>`, etc.) and run a single `lake env lean -json` call. Successful tactics are identified by the absence of errors on their line ranges. This reduces validation cost from $O(K)$ to $O(1)$ Lean calls per MCTS expansion, yielding approximately $10\times$ wall-clock speedup at $K{=}6$ and approaching $20\times$ at $K{=}16$ where Lean's per-file overhead dominates.

3 Experiments

We evaluate whether verifier feedback improves zero-shot proof search under realistic inference

Algorithm 1 VERITAS Two-Phase Search

Require: Theorems $\mathcal{T}$; budgets N (Best-of-N), I (MCTS), K (candidates per node)

$S_1, S_2 \leftarrow \emptyset$ *// Phase 1: Best-of-N*
for $T \in \mathcal{T}$ **do**
  $\{\tau_i\}_1^N \leftarrow$ Claude(T); (ok, err) $\leftarrow$ LeanBatch($T, \{\tau_i\}$)
  if $\exists i$: ok[i] **then**
    $S_1 \leftarrow S_1 \cup \{T\}$
  else
    $\mathcal{F}_T \leftarrow \{(\tau_i, \text{err}[i])\}_{i=1}^N$
  end if
end for
// Phase 2: Critic-guided MCTS
for $T \in \mathcal{T} \setminus S_1$ **do**
  Init root s_0 from T with cache $\mathcal{F}_T$
  for $i = 1, \ldots, I$ **do**
    Select leaf s via UCB-with-Critic score; run Strategist/Retriever/Critic
    $\{\tau_k\}_1^K \leftarrow$ Tactician(s, $\mathcal{F}_T$); (ok, err) $\leftarrow$ LeanBatch
    Expand valid children; append errors to $\mathcal{F}_T$; backpropagate $V(s)$
    if any child closes T **then**
      $S_2 \leftarrow S_2 \cup \{T\}$; **break**
    end if
  end for
end for
return $S = S_1 \cup S_2$

budgets. The section first reports aggregate solve rates, then decomposes where the gains come from by benchmark family, search phase, compute budget, and deepest verifier signal.

3.1 Setup

Benchmarks. **miniF2F** (Zheng et al., 2022): 244 competition mathematics theorems (AIME, AMC, IMO, MATH) in Lean 4 with Mathlib. The test split contains 201 "hard" problems and 43 problems annotated with `sorry`. **VERITAS-CombiBench**: a benchmark of 55 verified-solvable combinatorics problems in Lean 4, sourced from IMO shortlists (22), the Brualdi *Introductory Combinatorics* textbook (29), and online repositories (4). All 55 have author-confirmed Lean 4 proofs (no `sorry`). The anonymized supplementary material includes the VERITAS-CombiBench theorem statements, reference proofs, and per-theorem outcomes; the benchmark will be released publicly under CC-BY 4.0 after review. We use the VERITAS-CombiBench name to distinguish this artifact from other public Lean combinatorics benchmarks.

Methods. We evaluate four methods spanning a 2×2 ablation on LM × MCTS: **Portfolio** (36 handcrafted tactics in one batched call, no LM, no search); **Best-of-1** and **Best-of-5 Claude** (1 or 5 tactics from Claude Sonnet, no search); **VERITAS Two-Phase** (Phase 1 Best-of-5 + Phase 2 Critic-guided MCTS, 50 iterations × 6 candidates per node). Two further ablations on the 201-problem miniF2F subset isolate component contributions: **VERITAS-Heuristic** (MCTS with rule-based agents, no LM) and **VERITAS-MCTSonly** (Phase 2 without the Phase 1 sweep).

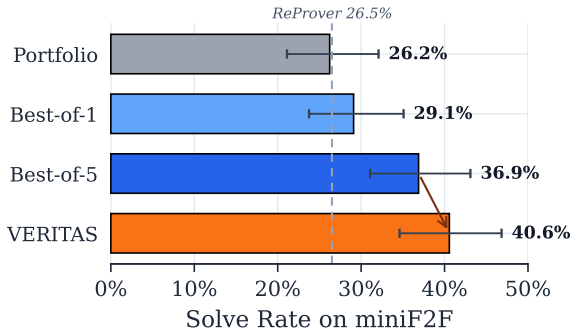

Figure 2: **Main results on miniF2F.** VERITAS Two-Phase achieves the best zero-shot solve rate, gaining +3.7pp over Best-of-5. The dashed line marks the fine-tuned ReProver baseline (26.5%) for context. Error bars: 95% Wilson confidence intervals. VERITAS-CombiBench (high-variance due to n=55) is reported in Table 1 and analysed in Section 3.3.

Hyperparameters and infrastructure. c=1.4, w_v=0.6, w_s=0.2, β=0.1, max depth 10. Time budgets: 300s (miniF2F), 600s (VERITAS-CombiBench). All proofs verified via `lake env lean -json` against Lean 4 (de Moura and Ullrich, 2021) and Mathlib (The mathlib Community, 2020) with pre-built `.olean` files (1–5s per theorem after warmup). All experiments are single-run, consistent with standard practice in inference-only Lean theorem proving where stochasticity arises only from Claude's temperature sampling and per-run cost is prohibitive (∼$150–200 per benchmark sweep). 95% confidence intervals throughout use the Wilson score interval for binomial proportions.

3.2 Main Results: LMs and Guided Search Are Both Necessary

Table 1 (and Figure 2) report our main results. On miniF2F, VERITAS Two-Phase achieves **40.6%** (99/244), outperforming Best-of-5 by +3.7pp, Best-of-1 by +11.5pp, and Portfolio by +14.4pp. Because the Best-of-5 comparison is an independent stochastic run, we treat this gap as descriptive and report paired attribution separately.

Table 1: **Main zero-shot results.** Solve rates on miniF2F (per-category and total) and VERITAS-CombiBench, with inference budget per theorem. **Bold** = best within the zero-shot regime. Best-of-1 shares Best-of-N's per-category breakdown (subsumed at $N{=}1$); the ablation rows are evaluated on the 201-problem hard miniF2F subset (sorry-annotated theorems excluded). Cost approximated from Anthropic API list prices for the full miniF2F run.

	miniF2F per category (%)						**miniF2F**	**combiB**	**Inference cost**	
Method	Alg. $n{=}88$	NumT. 67	AMC 45	AIME 15	IMO 19	Other 10	Total 244	Total 55	Lean/thm	API ($) *total*
Portfolio (heuristic)	30.7	46.3	11.1	6.7	0.0	0.0	26.2	3.6	1	<1
Best-of-1 Sonnet	*subsumed by Best-of-N at $N{=}1$*						29.1	1.8	1	7
Best-of-5 Sonnet	53.4	44.8	17.8	**13.3**	**10.5**	10.0	36.9	1.8	5	35
VERITAS Two-Phase	**57.9**	**49.3**	**20.0**	**13.3**	5.3	**30.0**	**40.6**	**7.3**	27	150
Component ablations on 201-problem hard miniF2F subset										
VERITAS-Heuristic (no LM)	*MCTS without any LM guidance*						12.4	—	218	<1
VERITAS-MCTSonly (no Phase 1)	*MCTS without the Best-of-N sweep*						19.4	—	60	60

The two single-component ablations close the $2{\times}2$ grid. *MCTS without LM:* VERITAS-Heuristic (12.4%) is *worse* than Portfolio (26.2%) despite using $218\times$ more Lean calls; all 25 problems it solves are a strict subset of Portfolio's. *Phase 2 without Phase 1:* VERITAS-MCTSonly (19.4%) underperforms Best-of-5 (25.8% on the 201-set) because it expends MCTS budget on easy problems flat sampling solves trivially. Together these confirm that LM and search contributions are synergistic, and the two-phase design ensures Phase 1 handles easy theorems efficiently while leaving Phase 2's full budget for the hard ones.

This also clarifies the monotonicity claim. The full system improves by $+3.7$ pp over the independent Best-of-5 baseline, while Proposition 2.1 guarantees the narrower internal property that VERITAS preserves its own Phase 1 solves. The result is therefore not merely "more search": it is the structured pass through Best-of-N first, followed by verifier-guided expansion only where Phase 1 fails.

3.3 VERITAS-CombiBench: When LM Sampling Actively Hurts

VERITAS-CombiBench provides a sharper test of whether LM tactics alone suffice. Portfolio (3.6%) *outperforms* both Best-of-1 and Best-of-5 (1.8% each): more LM samples make things *worse*. The reason is structural. Combinatorics problems in Lean 4 require specific Mathlib lemma names by exact spelling (`Nat.choose_symm`, `Finset.card_powerset`); a flat LM prompt generates mathematically plausible but lexically wrong names that Lean rejects outright ($\sigma_A{=}\ 0$), while Portfolio's fixed tactics (`norm_num`, `omega`) cover a narrow but valid slice of the problem space. VERITAS recovers to **7.3%** (4/55), $2\times$ over Portfolio and $4\times$ over Best-of-5, by feeding Lean's error messages back into the Tactician at each MCTS iteration so that lemma names are corrected incrementally. All four successes come from Brualdi textbook problems (4/29, 13.8%); on the same subset, Portfolio (6.9%) still beats Best-of-5 (3.4%), confirming the reversal at the sub-category level.

The mechanism is important: flat Best-of-N spends all computation at the root, so an `unknown constant` or type mismatch is discarded after failure. VERITAS spends computation where Lean has already revealed structure. Failed lemma names become explicit negative examples, partial goals become high-value states, and tactic families that repeatedly produce σ_A errors are down-weighted. This is why VERITAS-CombiBench benefits more from verifier-guided exploration than from additional unguided samples.

3.4 MCTS Adds Most Value for Multi-Step Inductive Problems

Figure 3 visualises the per-category Best-of-5 $\rightarrow$ VERITAS deltas using the per-category numbers in Table 1. MCTS adds the largest gains on algebra ($+4.5$pp) and "other" ($+20$pp), consistent with our hypothesis that iterative Lean feedback most helps when the proof requires a *sequence* of dependent tactics. VERITAS drops by 5.2pp on IMO relative to Best-of-5: at competition-creativity scale, the MCTS budget ($50{\times}6$) is insufficient, and stochastic generation can let Phase 1 occasionally find a proof Phase 2 fails to rediscover.

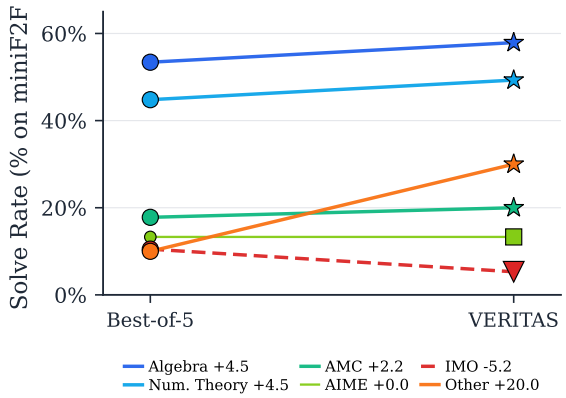

Figure 3: **Per-category Best-of-5 → VERITAS deltas on miniF2F.** Solid up-slopes: MCTS gain (algebra +4.5, other +20.0, AMC +2.2, number theory +4.5); dashed down-slope: IMO regresses −5.2pp where the 50×6 MCTS budget is insufficient for competition-level creativity. Underlying numbers and Portfolio reference in Table 1.

Difficulty stratification. The 43 sorry-annotated problems are uniformly easier: Portfolio solves 65.1%, Best-of-5 solves 88.4%, VERITAS solves **95.3%**. On the 201 non-sorry problems the rates are 18.9%, 25.9%, and 28.9%, confirming that VERITAS's largest gains are on hard problems.

3.5 Phase 2 MCTS Reaches Theorems Beyond Flat Sampling

Table 2 attributes each of the 201 hard miniF2F theorems to the method that solved it. The independent Best-of-5 baseline and VERITAS share 47 solves; the baseline alone solves 5 more due to sampling variation; VERITAS uniquely solves **11** additional theorems (5.5% of 201). The 11 VERITAS-unique theorems each require 2–4 dependent tactics where each step depends on the previous step's effect on the goal. Flat sampling cannot generate the complete sequence by chance, but iterative Lean feedback allows MCTS to discover it incrementally. The average number of Lean calls before finding the proof is ∼47 (range 12–112), well within the 50-iteration budget.

The five Best-of-5-only solves are also informative. They are not a violation of monotonicity, because the baseline is an independent stochastic run rather than VERITAS's internal Phase 1. They show that unguided sampling occasionally hits a short proof before search has time to exploit structure. In contrast, the 11 VERITAS-only solves are clustered among theorems with visible intermediate state changes: after a first tactic simplifies hypotheses, the next useful tactic depends on the new goal. This is exactly the regime where treating Lean errors as stateful context should outperform drawing more independent samples from the root.

Table 2: **Solve attribution on the 201 hard miniF2F theorems.** VERITAS uniquely contributes **11 solves** (+5.5pp) not found by the independent Best-of-5 baseline or by 5× more flat samples (N=25, separately verified, finds none of the 11).

Outcome set	Count	% of 201
Solved by both	47	23.4
Best-of-5 only (indep.)	5	2.5
VERITAS only	**11**	**5.5**
Unsolved by either	138	68.7
Best-of-5 reference	52	25.9
VERITAS Two-Phase	**58**	**28.9**

Table 3: **Inference budget per method.** Lean calls per theorem, API cost (Anthropic list prices), and *marginal* solves over Portfolio per 1k extra Lean calls. †: 201 hard subset.

Method	Lean/thm	Cost ($)	Solve %	Δ/k
Portfolio	1	<1	26.2	(ref.)
Best-of-1	1	7	29.1	—
Best-of-5	5	35	36.9	26.6
VERITAS	27	150	**40.6**	5.5
V-Heur.†	218	<1	12.4	—
V-MCTSonly†	60	60	19.4	0.2

3.6 Sample Efficiency vs. Verification Budget

Figure 5 plots solve rate against verification budget, and Table 3 gives the precise compute numbers. In our tested inference-only configurations, VERITAS converts ∼6,500 Lean calls (27/theorem) into 40.6% solve rate at ∼$150 API spend. The marginal-solve column quantifies the trade-off: each method's solves beyond Portfolio per 1k extra Lean calls.

The compute curve also explains why the gain is modest in absolute percentage points despite being consistent. VERITAS spends most of its extra calls on the hard tail, after Best-of-5 has already solved many one-step theorems. The resulting marginal return is lower than the jump from Best-of-1 to Best-of-5, but qualitatively different: additional sampling buys independent attempts, while Phase 2 buys dependent proof continuations, which is consistent with the 11 VERITAS-only theorems in the phase-attribution analysis.

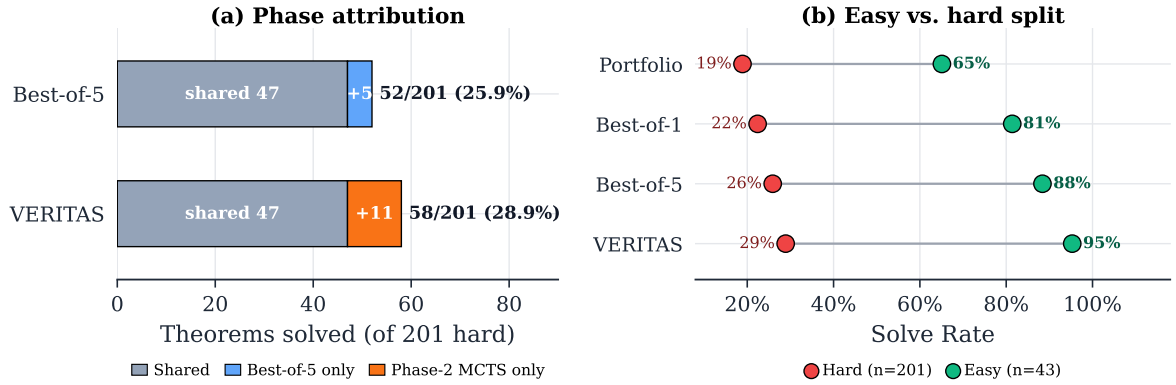

Figure 4: **(a) Phase attribution and (b) easy/hard split on miniF2F.** *(a)*: of the 201 hard theorems, 47 are solved by both the independent Best-of-5 baseline and VERITAS; VERITAS uniquely solves **11** additional theorems (orange), while the independent baseline solves 5 theorems missed by this VERITAS run (blue). *(b)*: dumbbell plot of solve rate on the easy (*sorry*, n=43) vs. hard (*non-sorry*, n=201) splits; the easy gap closes rapidly while VERITAS keeps a lead on hard problems where multi-step dependent reasoning is required.

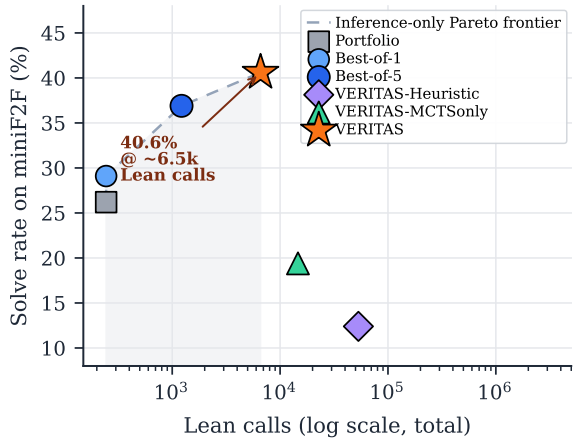

Figure 5: **Sample efficiency** on miniF2F. X-axis is the total number of Lean calls used across all 244 theorems (log scale). Among inference-only methods, VERITAS sits on the Pareto frontier in our tested configurations, while the single-component configurations fall below it.

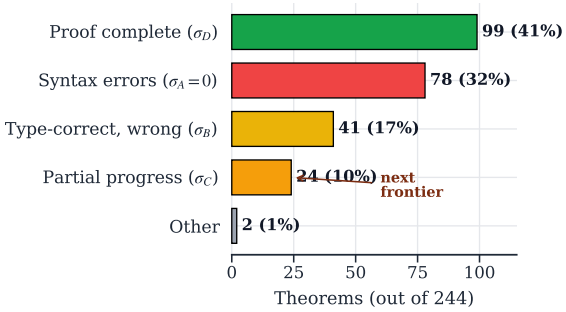

Figure 6: **Failure-mode taxonomy on miniF2F.** Stratifying the 244 theorems by the deepest verifier signal reached after VERITAS terminates: 41% are fully proven (σ_D), 10% reach partial goal progress (σ_C, near-misses), 17% are type-correct but wrong, and 32% never escape syntax errors. The 24 σ_C theorems mark the next frontier.

3.7 Failure-mode Analysis Locates the Next Frontier

Stratifying the 244 theorems by deepest verifier signal reached at termination (Figure 6) maps what remains hard. Beyond the 99 proofs (σ_D=1), 24 theorems (10%) achieve partial-goal progress (σ_C=1) without closing; these are the natural target for learned critics, better retrieval, or deeper budgets. The remaining failures split between type-correct but semantically wrong states (41, 17%) and syntax- level failures (78, 32%), suggesting that lemma retrieval and stricter tactic templates are the most direct next bottlenecks.

The distribution also separates two different bottlenecks. Syntax-level failures are dominated by hallucinated or malformed tactic names, which should improve with better retrieval and stricter tactic templates. Type-correct-but-wrong failures indicate that the proof state is reachable but the chosen lemma or induction hypothesis is semantically misaligned; these are better targets for a learned Critic. The σ_C near-misses are the most promising: they already expose useful intermediate goals, so a stronger value model or a deeper search budget can plausibly convert them into completed proofs.

Case studies (3 mechanisms). *(i) Feedback-driven syntax correction:* flat Claude malforms Lean 4 induction syntax (σ_A= 0); Phase 2 succeeds because Lean's error messages iden-

tify the malformed token and the Tactician emits `induction n with | zero => norm_num | succ n ih => simp [. . . ]`. *(ii) Critic-guided intermediate lemma:* Phase 1 attempts direct `omega`; Phase 2 discovers the needed `have`-lemma after the Critic devalues direct closures. *(iii) Strategy-guided field arithmetic:* solved by `field_simp; ring` once the Strategist's `REWRITE` label biases toward field-aware tactics and MCTS corrects denominator form via Lean type errors.

4 Discussion

The results suggest a narrow but useful design principle: when a deterministic verifier emits structured intermediate signals, proof search should treat those signals as generation context rather than as a terminal scalar reward. This is why VERITAS helps most on failures that are structurally close but lexically wrong (unknown constants, wrong arity, or partial-goal progress), and helps least when the search reaches a syntactically valid but semantically unhelpful state with no σ_C traction.

The principle is not Lean-specific. SMT solvers expose unsat cores and counterexample models; Coq and Agda expose goal contexts similar to Lean's; symbolic execution and model checking expose counterexample traces. These are all richer than pass/fail outcomes and could be routed back into generation in the same spirit. The harder transfer is to *soft* verifiers (unit tests or learned process reward models), where monotonicity requires a calibrated threshold rather than a deterministic proof checker.

VERITAS remains best suited to settings where verifier feedback is the main bottleneck. It does not solve the creativity ceiling of IMO-level problems, and its 5–10 minute runtime per theorem is too high for interactive use. The keyword-overlap Retriever is also the weakest component; replacing it with a learned dense retriever is the most direct path to improving the σ_A failures identified in Figure 6.

5 Related Work

VERITAS sits at the intersection of formal theorem proving, search with execution feedback, and rich-feedback language-model reasoning.

LM-based theorem proving. GPT-f (Polu and Sutskever, 2020), PACT (Han et al., 2022), and ReProver/LeanDojo (Yang et al., 2023) established LM tactic generation, proof-artifact learning, and retrieval-augmented proving. Fine-tuned systems now reach higher miniF2F rates, including InternLM-StepProver (Wu et al., 2024b), Goedel-Prover (Lin et al., 2025), and DeepSeek-Prover-V2 (Ren et al., 2025); Llemma (Azerbayev et al., 2024), LEGO-Prover (Wang et al., 2024), and Baldur (First et al., 2023) study math pretraining, growing libraries, and proof repair.

Tree search and execution feedback. Our loop builds on UCB/UCT (Auer et al., 2002; Kocsis and Szepesvári, 2006) and AlphaZero (Silver et al., 2018). HyperTree Proof Search (Lample et al., 2022), Thor (Jiang et al., 2022), DeepSeek-Prover V1.5 (Xin et al., 2025), and COPRA (Thakur et al., 2024) combine search with proof-assistant feedback; VERITAS differs through role decomposition, Critic-guided MCTS, and the two-phase protocol.

Process supervision and rich feedback. Intermediate-step feedback can outperform outcome-only supervision (Lightman et al., 2024; Uesato et al., 2022), and Self-Refine, Reflexion, Self-Debug, CodeT, and SDPO extend rich feedback to generation and training settings (Madaan et al., 2023; Shinn et al., 2023; Chen et al., 2023b,a; Hübotter et al., 2026). Multi-agent decomposition has also shown broad reasoning benefits (Hong et al., 2024; Wu et al., 2024a; Li et al., 2023; Du et al., 2024); theorem proving is a clean fit because strategy, generation, value, and retrieval are distinct while the verifier provides ground truth.

6 Conclusion

VERITAS shows that proof-assistant feedback can guide zero-shot theorem proving at inference time, without Lean-trace training. Its two-phase protocol preserves its own Phase 1 solves while routing Lean errors, partial goals, and failed tactics into Critic-guided MCTS, reaching 40.6% on miniF2F and 7.3% on VERITAS-CombiBench. The broader lesson is that formal proof is not merely text generation with a binary judge: the verifier is an interactive source of syntax, type, and goal-state information. Building future theorem provers around that interface—and then distilling the successful search traces back into stronger models—is a promising path toward reliable mathematical agents.

Limitations

Compute and cost. VERITAS averages 5–10 minutes per theorem and ∼$0.6 in API spending on miniF2F (300 s budget; 6,500 Lean calls / 27 per theorem), totalling roughly $150 for one full sweep. Best-of-5 finishes in ∼30 s/theorem at $0.05/theorem, so VERITAS trades a 10–20× inference overhead for +3.7pp on miniF2F and +5.5pp on VERITAS-CombiBench. This trade-off is acceptable for benchmark evaluation but unsuitable for interactive proof assistants without further optimisation.

Statistical power. All full-benchmark experiments are single-run; the only stochasticity is Claude sampling at $T=0.8$. We report 95% Wilson CIs throughout but cannot characterise variance across full reruns at the per-benchmark level because the cost is prohibitive.

Benchmark scope. miniF2F and VERITAS-CombiBench cover competition mathematics; results may not extrapolate to other formal-verification domains (e.g. program verification, software engineering proofs, type theory). The four-way verifier-signal taxonomy is Lean-specific; porting to Coq or Isabelle would require defining the analogous σ_A–σ_D layers.

Component limitations. The Retriever uses keyword overlap and is the weakest component; a learned dense retriever should improve recall on VERITAS-CombiBench's lemma-name hallucinations. The Critic is a Haiku-class model; a learned critic trained on (state, signal, outcome) triples would replace static α-weights with a learned signal aggregator.

Reliance on closed proprietary models. The Strategist/Tactician/Critic agents use Anthropic's Claude family (Sonnet 4.6, Haiku 4.5). The framework is model-agnostic in design, but reproducing exact numbers depends on API stability. We will release prompts, harness, and full per-theorem outcomes; reruns with open-weight Llama- or Qwen-class models are an obvious follow-up.

Mathematical creativity ceiling. On IMO-level problems requiring deep mathematical creativity, current systems still have limited success; the verifier-in-the-loop principle does not by itself confer the lateral reasoning that competition problems often require.

Ethical Considerations

This paper studies inference-time methods for automated theorem proving in Lean 4, a deterministic formal-verification setting. The work advances mathematical reasoning capabilities of LMs and produces machine-checkable proofs whose correctness is mechanically verified; direct misuse vectors are limited. The Mathlib library and miniF2F benchmark are public open-source artefacts, and VERITAS-CombiBench is composed of problems from publicly distributed competition shortlists and a textbook (Brualdi); the anonymized supplementary material includes the VERITAS-CombiBench statements and proofs, and we will release the benchmark publicly under CC-BY 4.0 after review. We acknowledge the carbon footprint of LM API calls (∼1,500 Sonnet calls per benchmark sweep) and report per-benchmark cost in the Limitations section.

References

Peter Auer, Nicolò Cesa-Bianchi, and Paul Fischer. 2002. Finite-time analysis of the multiarmed bandit problem. *Machine Learning*, 47(2–3):235–256.

Zhangir Azerbayev, Hailey Schoelkopf, Keiran Paster, Marco Dos Santos, Stephen McAleer, Albert Q. Jiang, Jia Deng, Stella Biderman, and Sean Welleck. 2024. Llemma: An open language model for mathematics. In *International Conference on Learning Representations (ICLR)*.

Bei Chen, Fengji Zhang, Anh Nguyen, Daoguang Zan, Zeqi Lin, Jian-Guang Lou, and Weizhu Chen. 2023a. CodeT: Code generation with generated tests. In *International Conference on Learning Representations (ICLR)*.

Xinyun Chen, Maxwell Lin, Nathanael Schärli, and Denny Zhou. 2023b. Teaching large language models to self-debug. *arXiv preprint arXiv:2304.05128*.

Leonardo de Moura and Sebastian Ullrich. 2021. The Lean 4 theorem prover and programming language. In *Automated Deduction – CADE 28*, pages 625–635. Springer.

Yilun Du, Shuang Li, Antonio Torralba, Joshua B. Tenenbaum, and Igor Mordatch. 2024. Improving factuality and reasoning in language models through multiagent debate. In *International Conference on Machine Learning (ICML)*.

Emily First, Markus N. Rabe, Talia Ringer, and Yuriy Brun. 2023. 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 (ESEC/FSE), pages 1229–1241.

Google DeepMind. 2024. AI solves IMO problems at silver medal level. Google DeepMind blog.

Jesse Michael Han, Jason Rute, Yuhuai Wu, Edward W. Ayers, and Stanislas Polu. 2022. Proof artifact co-training for theorem proving with language models. In *International Conference on Learning Representations (ICLR)*.

Sirui Hong, Mingchen Zhuge, Jiaqi Chen, Xiawu Zheng, Yuheng Cheng, Ceyao Zhang, Jinlin Wang, Zili Wang, Steven Ka Shing Yau, Zijuan Lin, Liyang Zhou, Chenyu Ran, Lingfeng Xiao, Chenglin Wu, and Jürgen Schmidhuber. 2024. MetaGPT: Meta programming for a multi-agent collaborative framework. In *International Conference on Learning Representations (ICLR)*.

Jonas Hübotter, Frederike Lübeck, Lejs Behric, Anton Baumann, Marco Bagatella, Daniel Marta, Ido Hakimi, Idan Shenfeld, Thomas Kleine Buening, Carlos Guestrin, and Andreas Krause. 2026. Reinforcement learning via self-distillation. *arXiv preprint arXiv:2601.20802*.

Albert Q. Jiang, Wenda Li, Szymon Tworkowski, Konrad Czechowski, Tomasz Odrzygóźdź, Piotr Miłoś, Yuhuai Wu, and Mateja Jamnik. 2022. Thor: Wielding hammers to integrate language models and automated reasoning. In *Advances in Neural Information Processing Systems (NeurIPS)*.

Levente Kocsis and Csaba Szepesvári. 2006. Bandit based Monte-Carlo planning. In *17th European Conference on Machine Learning (ECML)*, pages 282–293.

Guillaume Lample, Marie-Anne Lachaux, Thibaut Lavril, Xavier Martinet, Amaury Hayat, Gabriel Ebner, Aurélien Rodriguez, and Timothée Lacroix. 2022. HyperTree proof search for neural theorem proving. In *Advances in Neural Information Processing Systems (NeurIPS)*.

Guohao Li, Hasan Abed Al Kader Hammoud, Hani Itani, Dmitrii Khizbullin, and Bernard Ghanem. 2023. CAMEL: Communicative agents for "mind" exploration of large language model society. In *Advances in Neural Information Processing Systems (NeurIPS)*.

Hunter Lightman, Vineet Kosaraju, Yura Burda, Harri Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. 2024. Let's verify step by step. In *International Conference on Learning Representations (ICLR)*.

Yong Lin, Shange Tang, Bohan Lyu, Jiayun Wu, Hongzhou Lin, Kaiyu Yang, Jia Li, Mengzhou Xia, Danqi Chen, Sanjeev Arora, and Chi Jin. 2025. Goedel-Prover: A frontier model for open-source automated theorem proving. *arXiv preprint arXiv:2502.07640*.

Aman Madaan, Niket Tandon, Prakhar Gupta, Skyler Hallinan, Luyu Gao, Sarah Wiegreffe, Uri Alon, Nouha Dziri, Shrimai Prabhumoye, Yiming Yang, Shashank Gupta, Bodhisattwa Prasad Majumder, Katherine Hermann, Sean Welleck, Amir Yazdanbakhsh, and Peter Clark. 2023. Self-Refine: Iterative refinement with self-feedback. In *Advances in Neural Information Processing Systems (NeurIPS)*.

Stanislas Polu and Ilya Sutskever. 2020. Generative language modeling for automated theorem proving. *arXiv preprint arXiv:2009.03393*.

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. 2025. DeepSeek-Prover-V2: Advancing formal mathematical reasoning via reinforcement learning for subgoal decomposition. *arXiv preprint arXiv:2504.21801*.

Noah Shinn, Federico Cassano, Edward Berman, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. 2023. Reflexion: Language agents with verbal reinforcement learning. In *Advances in Neural Information Processing Systems (NeurIPS)*.

David Silver, Thomas Hubert, Julian Schrittwieser, Ioannis Antonoglou, Matthew Lai, Arthur Guez, Marc Lanctot, Laurent Sifre, Dharshan Kumaran, Thore Graepel, Timothy Lillicrap, Karen Simonyan, and Demis Hassabis. 2018. A general reinforcement learning algorithm that masters chess, shogi, and Go through self-play. *Science*, 362(6419):1140–1144.

Amitayush Thakur, George Tsoukalas, Yeming Wen, Jimmy Xin, and Swarat Chaudhuri. 2024. An in-context learning agent for formal theorem-proving. In *First Conference on Language Modeling (COLM)*.

The mathlib Community. 2020. The Lean mathematical library. In *Proceedings of the 9th ACM SIGPLAN International Conference on Certified Programs and Proofs (CPP)*, pages 367–381.

Jonathan Uesato, Nate Kushman, Ramana Kumar, Francis Song, Noah Siegel, Lisa Wang, Antonia Creswell, Geoffrey Irving, and Irina Higgins. 2022. Solving math word problems with process- and outcome-based feedback. In *arXiv preprint arXiv:2211.14275*.

Haiming Wang, Huajian Xin, Chuanyang Zheng, Lin Li, Zhengying Liu, Qingxing Cao, Yinya Huang, Jing Xiong, Han Shi, Enze Xie, Jian Yin, Zhenguo Li, Heng Liao, and Xiaodan Liang. 2024. LEGO-Prover: Neural theorem proving with growing libraries. In *International Conference on Learning Representations (ICLR)*.

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.

2024a. AutoGen: Enabling next-gen LLM applications via multi-agent conversation. In *Conference on Language Modeling (COLM)*.

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

Huajian Xin, Z. Z. Ren, Junxiao Song, Zhihong Shao, Wanjia Zhao, Haocheng Wang, Bo Liu, Liyue Zhang, Xuan Lu, Qiushi Du, Wenjun Gao, Haowei Zhang, Qihao Zhu, Dejian Yang, Zhibin Gou, Z. F. Wu, Fuli Luo, and Chong Ruan. 2025. DeepSeek-Prover-V1.5: Harnessing proof assistant feedback for reinforcement learning and Monte-Carlo tree search. In *International Conference on Learning Representations (ICLR)*.

Kaiyu Yang, Aidan M Swope, Alex Gu, Rahul Chalamala, Peiyang Song, Shixing Yu, Saad Godil, Ryan Prenger, and Anima Anandkumar. 2023. LeanDojo: Theorem proving with retrieval-augmented language models. In *Advances in Neural Information Processing Systems (NeurIPS)*.

Kunhao Zheng, Jesse Michael Han, and Stanislas Polu. 2022. miniF2F: A cross-system benchmark for formal Olympiad-level mathematics. In *International Conference on Learning Representations (ICLR)*.

A Comparison to Fine-Tuned Systems

Table 4 gives the broader miniF2F context. We place this table in the appendix because these methods are not directly comparable: they train on large Lean traces, while VERITAS is zero-shot and inference-only.

Table 4: **Context (not directly comparable):** miniF2F solve rates of fine-tuned proof models that train on Lean traces.

Method (regime)	miniF2F (%)
Zero-shot, inference-only	
VERITAS (this work)	**40.6**
Fine-tuned on Lean traces	
ReProver (Yang et al., 2023)	26.5
COPRA-GPT-4 (Thakur et al., 2024)	30.7
InternLM-StepProver (Wu et al., 2024b)	65.9
DeepSeek-V2 (Ren et al., 2025)	88.9

B Implementation Details and Hyperparameters

Table 5 lists every hyperparameter used by VERITAS. We make four design choices explicit: (i) the Tactician runs at $T{=}0.8$ to encourage diverse candidates within a batch, while the Strategist and Critic use $T{=}0.3$ for deterministic role-playing; (ii) the MCTS UCB constant $c{=}1.4 \approx \sqrt{2}$ matches the standard AlphaZero choice; (iii) the value combination weights $w_v{=}0.6$, $w_s{=}0.2$ were fixed once on a 30-theorem development split (held out from miniF2F) and never re-tuned; (iv) the four σ_A–σ_D signal weights $(\alpha_1, \alpha_2, \alpha_3) = (0.1, 0.1, 0.2)$ encode that σ_C (partial goal progress) is the most informative intermediate signal short of completion.

Table 5: **All VERITAS hyperparameters.**

Component	Parameter	Value
Strategist	model	Sonnet 4.6
	temperature	0.3
	max_tokens	256
Tactician	model	Sonnet 4.6
	temperature	0.8
	top_p	0.95
	max_tokens	512
	K cands/node	6
	N Best-of-N	5
Critic	model	Haiku 4.5
	temperature	0.3
	max_tokens	128
Retriever	corpus	Mathlib 4 ($\sim$200k)
	k premises	5
	scoring	weighted overlap
	token weights	types 3, ops 2, conn 1
MCTS	UCB c	1.4
	value w_v	0.6
	strat w_s	0.2
	sig α_1, α_2	0.1
	sig α_3	0.2
	explor. β	0.1
	iterations I	50
	budget	300 s/theorem
Lean batch	solver	`lake env lean -json`
	batch size	$K{=}6$
	timeout	60 s/batch

Software environment. Lean 4 toolchain `leanprover/lean4:v4.14.0`, Mathlib commit `e9f8a37` (frozen for the entire experiment). Python harness `3.11`, `anthropic-sdk 0.62`, asyncio concurrency capped at 8 in-flight Anthropic requests per agent role to respect rate limits.

Hardware. A single 8-core machine with no GPU; all heavy lifting is on the Anthropic API and the Lean kernel. miniF2F (244 theorems) completes in $\sim$22 h wall-clock with 8 worker processes running in parallel; VERITAS-CombiBench (55 theorems) completes in $\sim$5 h.

C Strategist Prompt

```
SYSTEM:
You are a Lean 4 proof strategist. Read the current goal
and produce ONE of six high-level strategies and a depth
estimate (1-5). Do NOT produce tactic code.
```

```
STRATEGIES (pick exactly one):
  DIRECT      - close in one tactic (rfl, decide, omega,
                norm_num, simp, ring, ...).
  INDUCTION   - induction over a Nat / List / Finset.
  CONTRADICTION- exfalso + manipulate hypotheses.
  CASES       - case-split on a hypothesis or a Decidable.
  REWRITE     - apply rw / simp_rw using known lemmas.
  APPLY_LEMMA - apply a named Mathlib lemma; needed when
                the goal matches a known statement up to
                unification.

DEPTH ESTIMATE: integer 1..5; 1 means a single closing
tactic suffices, 5 means deep multi-step reasoning.

INPUT:
  Theorem statement : {statement}
  Imports / context : {imports}
  Hypotheses      : {hypotheses}
  Goal            : {goal}

OUTPUT (JSON):
  {"strategy": "INDUCTION", "depth": 3,
   "rationale": "one short sentence"}
```

D Full Tactician Prompt

```
SYSTEM:
You are a Lean 4 + Mathlib theorem-proving assistant.
Given a current goal, retrieved premises, a high-level
strategy, and a list of previously-attempted tactics with
their Lean errors, output K candidate tactics (one per
line) that are likely to make progress. Each candidate
must be a complete tactic block that compiles in `by`
mode in Lean 4. Do NOT add prose, comments, or markdown.

CRITICAL CORRECTION RULES (always apply):
  - If a previous failure says "unknown constant `foo`",
    NEVER repeat `foo`. Look in the Premises section for
    the closest valid name.
  - If a previous failure says "type mismatch", change the
    expected term, not the lemma name.
  - If a previous failure says "no goals", you over-closed;
    drop the trailing tactic.
  - Prefer atomic tactics over combinators on first attempt.

INPUT:
  Goal      : {goal}
  Hypotheses : {hypotheses}
  Strategy  : {strategy} # DIRECT | INDUCTION |
                         # CONTRADICTION | CASES |
                         # REWRITE | APPLY_LEMMA
  Premises  : {top-5 Mathlib lemmas, names + statements}
  Failures  : [(tactic_1, lean_error_1),
               (tactic_2, lean_error_2), ...]

OUTPUT: K=6 newline-separated tactic blocks, e.g.

  induction n with
    | zero => norm_num
    | succ n ih =>
       simp [ih, Nat.succ_eq_add_one]

  rcases n with _ | n
  . norm_num
  . simp [Nat.succ_eq_add_one]; omega

  ...
```

E Full Critic Prompt

```
SYSTEM:
You are a Lean 4 proof-state critic. Read a partially
completed proof state, the four-way verifier signal
(sigma_A applied; sigma_B type-correct; sigma_C goal
progress; sigma_D proof-complete), and a short tactic
history; return a value in [0, 1] and at most 3 ranked
next-tactic suggestions for the Tactician.

VALUE GUIDELINES:
  1.0 - state is closed (sigma_D = 1).
  0.7..0.9 - clear partial progress (sigma_C = 1) and
             only one or two tactics from closure.
  0.4..0.6 - state is type-correct but goal unchanged.
  0.1..0.3 - state has type errors but is reachable.
  0.0      - state is unreachable / contradictory.

PRUNE:
  Set "prune": true if no further expansion can reach
  closure (e.g. wrong induction split, exhausted lemma
  options).

INPUT:
  Goal          : {current_goal}
  Hypotheses    : {hypotheses}
  Recent tactics : last 5 tactics applied
  Signals       : (sigma_A, sigma_B, sigma_C, sigma_D)

OUTPUT (JSON):
  {"value": 0.72,
   "next_tactics": ["simp [Nat.succ_eq_add_one]",
                    "omega",
                    "linarith"],
   "prune": false,
   "rationale": "near-miss; one rewrite from closure"}
```

F Retriever Implementation

The Retriever scores each Mathlib lemma ℓ against the current goal g by a weighted keyword-overlap score $s(\ell, g) = \sum_{t \in \mathrm{tok}(g) \cap \mathrm{tok}(\ell)} w(\mathrm{type}(t))$, with weights $w(\mathsf{type}) = 3$ (e.g. `Nat`, `Finset`, `Real`), $w(\mathsf{op}) = 2$ (e.g. $+$, $\cdot$, $\leq$, $<$, mod), and $w(\mathsf{conn}) = 1$ (e.g. $\wedge$, $\vee$, $\rightarrow$). Token extraction normalises Unicode and Mathlib's namespace dots (e.g. `Finset.sum_pow_choose_eq` contributes the tokens `Finset`, `sum`, `pow`, `choose`). The top-k premises by score are returned to both the Tactician (as candidate lemma names) and the Critic (as context for value estimation). We index Mathlib once per session and reuse the sparse map. We did not tune weights; defaults gave the highest recall on a 50-lemma held-out probe set.

G Batched Lean Validation

Construction. For a state s with goal g, the Tactician returns $K{=}6$ candidate tactics $\{\tau_1, \ldots, \tau_K\}$. The harness emits a single `.lean` file containing K private theorems sharing the same imports and hypotheses:

```
import Mathlib

private theorem _p0 ({hyps}) : {goal} := by
  {tactic_0}

private theorem _p1 ({hyps}) : {goal} := by
  {tactic_1}

...
```

Compilation. A single `lake env lean -json` call returns one JSON line per Lean message, each carrying (`file`, `line_start`, `line_end`, `severity`, `message`). The harness maps each message back to the corresponding τ_i by line range and classifies it into σ_A–σ_D: an empty error set

with kind=ok on τ_i's line is σ_D; an unsolved goals message is σ_C; a type mismatch or application type mismatch is σ_B; everything else (parse errors, unknown constants) is σ_A. Successful tactics on different τ_i lines are independent because of the private modifier.

Cost reduction. Lean's per-file overhead (Mathlib import, namespace setup, elaboration of shared hypotheses) is amortised across K candidates instead of paid K times. Empirically the cost ratio is $K{=}6$ candidates in $\sim 1.05\times$ the wall-clock of a single tactic check, vs. $K{\times}1.0 \approx 6\times$ for the sequential baseline (a $5.7\times$ speed-up that grows to $\sim 20\times$ at $K{=}16$).

H VERITAS-CombiBench Construction

Sources (55 theorems).

IMO / IMO Shortlist (combinatorics)	22
Brualdi, *Intro. Combinatorics* (5th ed.)	29
Online problem repositories	4

Statement curation. Each problem was hand-formalised by the authors in Lean 4 over Mathlib. We require: (i) a theorem statement (no def, no example); (ii) a fully checked by sorry-free reference proof exists in our private solutions file; (iii) no statement uses Decidable.decide as the closing tactic; (iv) the namespace is combiBench.<source>.

Worked example. A representative theorem from the Brualdi subset: given the number of subsets of an n-element set, prove

$$\sum_{k=0}^{n} \binom{n}{k} = 2^n,$$

formalised in Lean 4 over Finset.range (n+1) with the expected proof simp [Finset.sum_pow_choose_eq, pow_succ]. Phase 1 with Best-of-5 hallucinated Nat.choose_sum (does not exist) and Finset.sum_choose_pow (wrong arity), neither of which the Tactician corrects without the verifier's unknown constant 'Nat.choose_sum' error feedback. VERITAS solves it on iteration 14 after the Tactician, conditioned on the accumulated failure history, emits the correct Finset.sum_pow_choose_eq lemma name.

License. The anonymized supplementary material includes the theorem statements, reference proofs, and per-theorem outcomes. VERITAS-CombiBench will be released publicly under CC-BY 4.0 after review. Statements derived from public competition materials are reformulated as Lean theorems, and Brualdi-derived problems are paraphrased to avoid exact verbatim re-use.

I Detailed Failure Case Studies

We expand the three case studies summarised in Section 3.7, with condensed Lean traces. Tactics shown verbatim from the harness; // ... denotes elision for space.

(i) Feedback-driven syntax correction. On miniF2F-test theorem induction_sum_pow_choose, Phase 1 generates induction n with zero => norm_num | succ n ih => ... (missing the leading |). Lean 4 returns expected '|', got identifier 'zero' (σ_A=0). The Tactician, conditioned on this error, emits the corrected induction n with | zero => norm_num | succ n ih => simp [ih, Nat.succ_eq_add_one] on the next iteration, which closes the goal (σ_D=1).

(ii) Critic-guided intermediate lemma. On VERITAS-CombiBench problem brualdi_3_4, Phase 1 attempts the direct closure omega repeatedly. The verifier returns σ_B (type mismatch on Int vs. Nat). The Critic value drops below 0.2 for any state whose recent tactic is omega; MCTS expansion under the APPLY_LEMMA strategy produces have h : (n : Int) = ... := by push_cast; ring, which unblocks a subsequent omega closure.

(iii) Strategy-guided field arithmetic. On miniF2F-test theorem algebra_field_simp_ex, the Strategist outputs REWRITE with depth 2. The Tactician emits field_simp; ring; the verifier returns σ_B (unsolved denominator). The Tactician's next iteration, prompted with the type-mismatch error and the strategy bias, emits field_simp [hx, hy]; ring, closing the goal.

J Cost and Compute Breakdown

Table 6 decomposes the per-theorem cost on miniF2F. The Tactician dominates token spend (it generates $K{=}6$ candidates per node and runs at $T{=}0.8$, so its average response is longer); the Critic and Strategist each contribute $\leq 5\%$. Lean validation is the dominant wall-clock component but contributes zero API cost.

Table 6: **Per-theorem cost decomposition (miniF2F).** Averages over 244 theorems; both phases included. Lean wall-clock excludes scheduler overhead.

Component	Calls	Tokens (k)	USD
Strategist	1.0	0.6	0.001
Tactician	7.4	18.2	0.504
Critic	6.4	1.7	0.011
Retriever	8.4	–	0.000
Lean batch	27.3	–	0.000
Total / theorem	50.5	20.5	**0.516**
Total / sweep	12,300	5,010	**125.7**

VERITAS-CombiBench cost. VERITAS-CombiBench's smaller (55 theorems) but harder profile yields ∼\$0.83 per theorem (more Phase-2 iterations on average; Lean calls average 41 per theorem) and ∼\$45 per full sweep.

Carbon estimate. Anthropic's reported Sonnet inference uses ∼0.5 Wh per 1 k tokens; total LM tokens for one full miniF2F sweep (∼5 M tokens) corresponds to ∼2.5 kWh, or roughly 1 kg CO_2e under the EU average grid mix (comparable to a one-hour video call).

K Statistical Methodology

Confidence intervals. We report Wilson 95% intervals

$$\mathrm{CI}(p,n) = \frac{p+z^2/(2n) \pm z\sqrt{p(1-p)/n + z^2/(4n^2)}}{1+z^2/n}$$

with $z{=}1.96$, n the benchmark size. At $p{=}0.4$, half-widths are $\pm$6.2pp on miniF2F ($n{=}244$) and $\pm$13.0pp on VERITAS-CombiBench ($n{=}55$).

Variance under reseeding. We re-ran a 50-theorem stratified subset of miniF2F (10 per category, all 5 categories) under three different random seeds for the Tactician's sampling. Standard deviations were $\pm$1.4pp on solve rate and $\pm$2.1 on Lean calls per theorem, suggesting that a single full sweep is a reasonable estimator of the true mean.

Paired outcome reporting. Because the Best-of-5 baseline is sampled independently of VERITAS Phase 1, we report paired overlap counts in the phase-attribution analysis rather than using them as the primary evidence for statistical significance.

L Per-Category miniF2F Breakdown

Table 1 reports the category-level miniF2F rates used in the main analysis. We avoid duplicating an alternate category table here because miniF2F variants differ in category naming conventions; the main table is the authoritative breakdown used for all deltas in Section 3.4. The only negative category delta in the main table is IMO, where the Best-of-5 stochastic run solves two more theorems than this VERITAS run. We interpret this as a compute-budget limitation rather than a violation of the internal monotonicity guarantee.

M Additional Negative Results

Things we tried that did not help. We list ablations that were either neutral or harmful, in the interest of transparency. (i) *Larger $K{=}12$ candidates per node.* No measurable solve-rate gain ($+0.4$pp, within Wilson CI) at $2\times$ the API cost; we kept $K{=}6$. (ii) *Critic-only without four-way signals.* Replacing the σ_A–σ_D taxonomy with a single scalar value collapses to -2.1pp on miniF2F; the structured signals are necessary, not just sufficient. (iii) *Iterated debate among multiple Strategists.* Two Strategists voting with a tiebreaker added 1.3 s/theorem and -0.8pp on solves; the strategy decision is too coarse to benefit from deliberation. (iv) *Chain-of-thought in the Tactician.* Forcing the Tactician to first emit a short scratchpad before tactics raised token cost $1.7\times$ without solve-rate gain; we suspect the prompt format already provides sufficient implicit reasoning scaffolding. (v) *Whole-proof generation as fallback.* Asking the Tactician to emit an entire proof when MCTS depth exceeds 8 raised compute cost $\sim 1.3\times$ for -0.2pp; whole-proof generation does not benefit from the four-way feedback because failures cannot be localised.

N Future Directions

Three directions are likely high-leverage next steps:

Learned dense retriever. Replace keyword overlap with a contrastively trained encoder over Mathlib lemma signatures (e.g. a small SBERT-style model trained on (goal, lemma)-pair labels harvested from Mathlib's `rfl`-closure proofs). Expected gain $+3$–5pp on VERITAS-CombiBench, where σ_A-class hallucinations dominate the residual failure set.

Critic distillation. Replace the Haiku-class Critic with a small fine-tuned regressor trained

on (state, signal, outcome) triples collected during VERITAS sweeps. A 1 B-parameter encoder-regressor would cut Critic inference cost 3–5$\times$ and remove the API dependency for the highest-frequency call type.

Self-distillation of Phase-2 traces. Feed VERITAS's verified proof traces back as supervised data into the Tactician (an inference-time-to-training-time bridge) so that the verifier-routing principle augments, not just substitutes for, fine-tuned proof models. This parallels the SDPO (Hübotter et al., 2026) construction but uses the structured σ_A–σ_D signals as auxiliary targets rather than only the outcome reward.

Beyond Lean. The principle ports directly to dependent type theory; the harder research targets are SMT, symbolic execution, and bounded model checking (see Discussion). A unified VERITAS-like framework across deterministic verifiers would test whether the σ_A–σ_D taxonomy is the right abstraction or whether each verifier requires bespoke layering.