DOES SMALL LANGUAGE MODELS ACHIEVE SCIENTIFIC REASONING VIA VERIFICATION-AUGMENTED REASONING?

Anonymous authors
Paper under double-blind review

ABSTRACT

Small Language Models (SLMs) are a constrained subset of Large Language Models (LLMs) and offer a promising pathway toward deployable models, particularly for scientific reasoning in resource-limited and real-time settings. When integrated with Lean-based formal reasoning, where scientific reasoning is expressed as formal proofs in Lean 4, Coq, and Isabelle, and counterexample-based falsification, where incorrect hypotheses are falsified by admissible counterexamples, their limited capacity raises fundamental gaps in reliable scientific reasoning. Prior works have attempted to address these gaps *via* Chain-of-Thought (CoT), Self-Consistency, and Tree-of-Thought (ToT), as well as benchmark-driven evaluations including GSM8K and ProofNet, but offer *only* partial solutions. Recent benchmarks show that even State-of-the-Art (SOTA) SLMs produce incorrect intermediate reasoning steps on challenging problems, while Lean-based formal reasoning and counterexample-based falsification remain underexplored. To address this gap, we benchmark whether verification-augmented reasoning can compensate for limited model capacity. As no suitable dataset exists for this setting, we construct LEANDATA by extending SCIBENCH *via* the Draft-Sketch-Prove paradigm to translate natural-language scientific hypotheses in 10 domains with Lean-verified proofs and generate admissible counterexamples. We use LEANDATA to train five SLMs (1B-3B parameters) for Lean 4 tactic generation within a neuro-symbolic loop, where the model proposes tactics and the proof assistant provides deterministic execution feedback via reinforcement learning with rule-based rewards. Empirically, +RL+Verification improves few-shot E2E success from $26.8\% - 41.9\%$ to $47.3\% - 58.9\%$ and the average across VA, FR, SC, AC, and E2E from $33.8\% - 46.5\%$ to $51.4\% - 61.6\%$, with VA reaching 48.6%.

1 INTRODUCTION

Small Language Models (SLMs) are a constrained subset of Large Language Models (LLMs) that operate under strict parameters and compute Belcak et al. (2025), making them suitable for deployment in real-world Lin et al. (2025) and resource-limited environments Wang et al. (2025). In contrast to large-scale models, which rely on extensive parameters and compute Sui et al. (2025), SLMs aim to approximate reasoning capabilities with significantly reduced capacity, often on the order of 10^9 parameters or fewer Sharma & Mehta (2025). This constraint introduces a fundamental trade-off between compute and reasoning capability, particularly in scientific domains involving multi-step derivations Song et al. (2025), symbolic manipulation Kusumegi et al. (2025), and hypothesis validation Zhang et al. (2025). Therefore, understanding how SLMs can reliably perform scientific reasoning under such constraints remains an important and largely unexplored gap, particularly in resource-limited and real-time settings involving Lean-based formal reasoning Moura & Ullrich (2021) and counterexample-based falsification Li et al. (2026). Lean-based formal reasoning expresses scientific reasoning as formal proofs in systems such as Lean 4 Moura & Ullrich (2021), Coq Bertot & Castéran (2013), and Isabelle Nipkow et al. (2002), while counterexample-based falsification identifies admissible instances that violate incorrect hypotheses; both impose strict constraints on formal validity. Formal proof assistants encode reasoning as a sequence of valid transformations, i.e., $f_\theta(x) \to \pi$, where π denotes a proof trace satisfying formal constraints, ensuring step-

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.

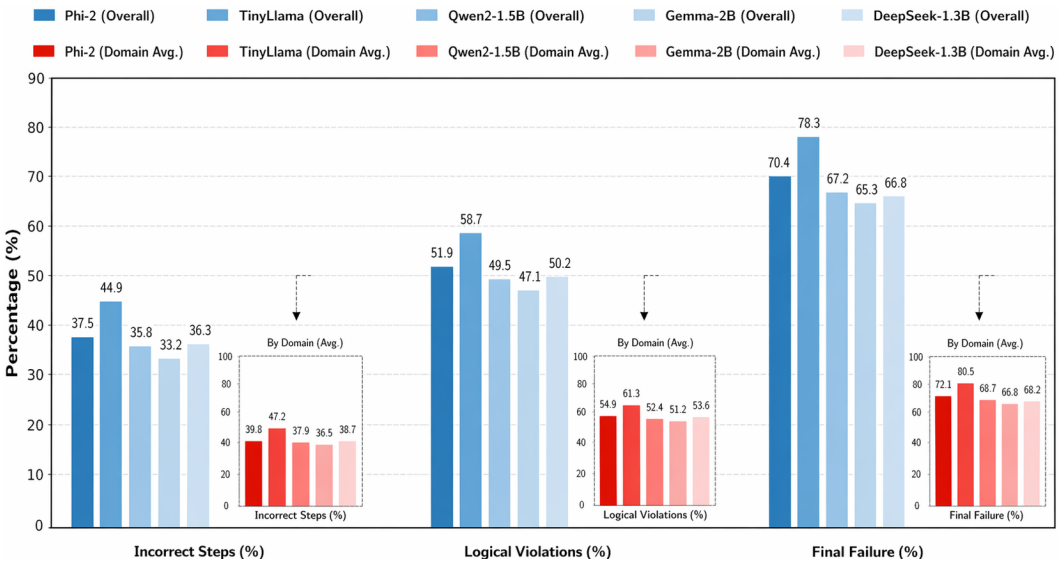


Figure 1: Empirical analysis of incorrect intermediate reasoning steps $r = (r_1, \ldots, r_k)$ in SLMs, motivating formal validity constraints $\pi \in \mathcal{P}_{valid}$ and counterexample-based falsification ($\exists x' : \neg\phi(x')$).

wise formal validity rather than heuristic reasoning Jiang et al. (2022), while counterexample-based falsification identifies instances x' such that a proposed hypothesis $h(x')$ fails. Rather than assuming that SLMs fundamentally lack reasoning capability, we focus on whether their reasoning processes can be made formally verifiable under formal constraints while supporting counterexample-based falsification through admissible instances x' where $h(x')$ fails.

Prior works have attempted to address these gaps *via* Chain-of-Thought (CoT) Wei et al. (2022), Self-Consistency Wang et al. (2022), and Tree-of-Thought (ToT) Yao et al. (2023), which improve reasoning by decomposing it into intermediate steps and modeling it as a sequence of latent decisions, often approximating $p(y|x)$ through reasoning steps $r = (r_1, \ldots, r_k)$. However, these works rely on heuristic intermediate steps that are not guaranteed to be formally valid, while benchmark-driven evaluations such as GSM8K Cobbe et al. (2021) and ProofNet Azerbayev et al. (2023) primarily focus on final-step correctness rather than enforcing intermediate reasoning validity. Recent benchmarks (e.g., Valmeekam et al. (2022)) further show that even State-of-the-Art (SOTA) SLMs produce incorrect intermediate reasoning steps on challenging problems, while Lean-based formal reasoning Moura & Ullrich (2021) and counterexample-based falsification Li et al. (2026) remain underexplored, highlighting a critical gap in formally verifiable scientific reasoning (see Figure 1).

To address this gap, we benchmark whether verification-augmented reasoning can provide process-level evidence of formal validity in scientific problem solving by constraining generated proof trajectories to executable validity constraints, i.e., $\pi \in \mathcal{P}_{valid}$, while recognizing that formal proof validity does not by itself guarantee semantic correctness of the natural-language reasoning. As no suitable dataset exists for this setting (see Table 1), we construct LEANDATA by extending SCIBENCH Wang et al. (2023) *via* the Draft-Sketch-Prove paradigm Jiang et al. (2022) to translate scientific problems across 10 domains into structured reasoning r, assumptions $\mathcal{H}(x)$, encoded formal specifications $\phi(x)$, Lean-verified proofs, and admissible counterexamples. We introduce LEANBENCH to evaluate five SLMs (1B-3B parameters) within a closed neuro-symbolic loop, where proof-assistant execution feedback guides reinforcement-learning-based tactic generation through the policy $\pi_\theta(a_t \mid s_t)$. In summary, the main contributions of this work are as follows:

- We construct LEANDATA, a 580-instance, 10-domain benchmark that jointly represents scientific reasoning r, assumptions $\mathcal{H}(x)$, encoded formal specifications $\phi(x)$, Lean-verified proofs $\pi \in \mathcal{P}_{valid}$, and admissible counterexamples x', and introduce LEANBENCH for evaluating five SLMs (1B-3B) in a Lean 4 neuro-symbolic loop.
- We show that +RL+Verification improves few-shot E2E success from $26.8\% - 41.9\%$ to $47.3\% - 58.9\%$ and the average across VA, FR, SC, AC, and E2E from $33.8\% - 46.5\%$ to $51.4\% - 61.6\%$, with VA reaching 48.6%.

2 RELATED WORKS

SLMs for General Reasoning. As discussed in §1, prior work improves SLM reasoning through structured prompting, data curation, and training strategies, with models, such as Phi-2 Javaheripi et al. (2023), TinyLlama Zhang et al. (2024), and DistilGPT Sanh et al. (2019) demonstrating reasoning under limited parameters and compute. Instruction tuning Ouyang et al. (2022), synthetic data distillation Gu et al. (2024), and reasoning-focused fine-tuning Lightman et al. (2024) improve arithmetic and symbolic reasoning, while Program-of-Thought (PoT) Chen et al. (2022) and tool-augmented generation Schick et al. (2023) use executable intermediate representations. Self-refinement Madaan et al. (2023), critique-based decoding Asai et al. (2024), and verifier-guided reranking Abdallah et al. (2025) further improve reliability by revising or filtering outputs. However, these works rely on learned, heuristic, or execution-based feedback and do not include formal validity of complete reasoning processes or guarantee that generated specifications satisfy executable logical constraints.

SLMs for Formal Reasoning. As discussed in §1, prior work trains language models to interact with proof assistants through tactic generation, structured supervision, and formal reasoning environments, such as LeanDojo Yang et al. (2023), miniF2F Zheng et al. (2021), and ProofNet Azerbayev et al. (2023) provide formal statements and proof traces for supervised and retrieval-augmented theorem proving, typically modeling tactic generation as $\pi_\theta(a_t \mid s_t)$ conditioned on proof states. Neuro-symbolic works such as Lean Copilot Song et al. (2024) further support iterative proof-assistant interaction through execution feedback, while AlphaProof Hubert et al. (2026) and DeepSeek-Prover Xin et al. (2025) advance formal theorem proving through large-scale training and synthetic Lean 4 data. The Draft-Sketch-Prove paradigm Jiang et al. (2022) bridges informal reasoning and formal proofs by progressively constructing proof sketches and executable proofs. However, these works remain predominantly focused on formal mathematics, with limited scientific coverage, and generally optimize proof construction rather than jointly modeling reasoning r, assumptions $\mathcal{H}(x)$, formal specifications $\phi(x)$, and counterexample-based falsification within a unified evaluation framework.

SLMs for Counterexample-Based and Falsification Reasoning. As discussed in §1, counterexample generation supports formal verification by seeking admissible inputs that violate a hypothesis or encoded formal claim, such as CounterMATH Li et al. (2025) and Learning to Disprove Li et al. (2026) explore counterexample-driven reasoning and formally verifiable disproof, but primarily target mathematical reasoning and do not integrate counterexample-based falsification with structured scientific reasoning and formal verification in a unified benchmark. Our LEANDATA and LEANBENCH address this gap by unified representing reasoning r, assumptions $\mathcal{H}(x)$, encoded formal specifications $\phi(x)$, Lean-verified proofs $\pi \in \mathcal{P}_{valid}$, and admissible counterexamples x' for SLM evaluation in a closed neuro-symbolic loop spanning formal verification and counterexample-based falsification.

3 METHODOLOGY

We formalize verification-augmented scientific reasoning in SLMs as a neuro-symbolic process that translates natural-language problems $x \in \mathcal{X}$ into encoded formal specifications $\phi(x)$ within a proof assistant while supporting counterexample-based falsification. Our pipeline consists of (i) LEANDATA, constructed by formalizing scientific reasoning into Lean-verified proofs and admissible counterexamples, and (ii) LEANBENCH, a benchmarking framework that evaluates SLMs within a loop spanning formal verification and counterexample-based falsification. An overview of the pipeline is shown in Figure 2.

3.1 LEAN DATASET

We construct LEANDATA, a dataset of 580 formalized scientific reasoning problems designed for formal verification and counterexample-based analysis. Each instance begins with a natural-language scientific problem $x \in \mathcal{X}$ from physics, chemistry, probability, calculus, differential equations, statistics, thermodynamics, quantum mechanics, geometry, or linear algebra, sourced from prior datasets such as SCIBENCH Wang et al. (2023). We extend the Draft-Sketch-Prove paradigm Jiang et al. (2022) to map x into an encoded formal specification $\phi(x)$ that can be formally ver-

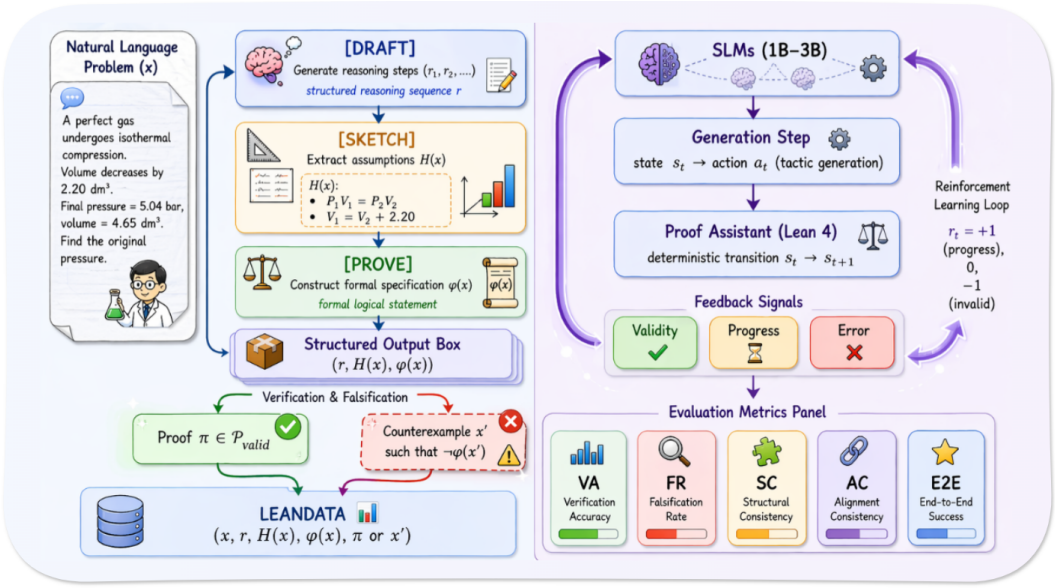


Figure 2: Overview of the pipeline: LEANDATA translates natural-language problems x to encoded formal specifications $\phi(x)$ via Draft-Sketch-Prove, producing Lean-verified proofs π or admissible counterexamples x', while LEANBENCH evaluates five SLMs (1B-3B) in a neuro-symbolic loop with proof-assistant execution feedback.

ified or falsified within a proof assistant by incorporating[1] (i) extraction of domain-specific equations and assumptions, (ii) normalization of variables and units, and (iii) support for formal verification and counterexample-based falsification. Given x, the pipeline produces reasoning steps $r = (r_1, \ldots, r_k)$, intermediate equations and assumptions $\mathcal{H}(x) = \{h_1, \ldots, h_m\}$, and an encoded formal specification $\phi(x)$ expressed in Lean, as shown in Equation (1).

$$x \xrightarrow{\text{DSP}} \big(r, \mathcal{H}(x), \phi(x)\big) \tag{1}$$

The mapping in Equation (1) is implemented using a 1B-3B-parameter SLM, which generates each component through stage-specific prompting (see Figure 3). We validate $(r, \mathcal{H}(x), \phi(x))$ through staged structural, alignment, and formal verification checks, with failed outputs refined through re-prompting up to K attempts ($K = 3$; see §6) and discarded if consistency is not achieved. Formal verification establishes that $\phi(x)$ is provable in Lean through a valid proof trace $\pi = (a_1, \ldots, a_T)$ under deterministic transitions $s_{t+1} = T(s_t, a_t)$, such that $\pi \in \mathcal{P}_{valid}$, but does not establish the semantic correctness of natural-language reasoning r or the faithfulness of extracted assumptions $\mathcal{H}(x)$; structural consistency, cross-stage alignment, and semantic faithfulness are therefore evaluated separately (see §6). We assess the semantic faithfulness of the resulting formal specifications through a human audit of 200 randomly sampled problems spanning all ten scientific domains (see Table 5 and §6).

To support counterexample-based falsification, let $h_x : \mathcal{X} \to \text{Prop}$ denote the scientific hypothesis or claim encoded by the formal specification $\phi(x)$. For falsification instances, we construct candidate inputs $x' \in \mathcal{X}$ through controlled variations over variables, parameters, and boundary conditions, modifying constraints and instantiating variables within valid domains. These variations provide candidate counterexamples, while the model is required to identify a valid falsifying input. A candidate x' is accepted only when it is admissible under the domain constraints and satisfies $\neg h_x(x')$ under formal checking; candidates violating domain constraints or producing invalid expressions are filtered, with generation repeated up to K attempts ($K = 3$; see §6) and invalid instances discarded. Counterexample generation is restricted to parameterized hypotheses, inequalities, constraints, or scientific relationships that admit valid falsifying instances, while problems with a unique numerical solution are evaluated exclusively through formal verification. The resulting dataset consists of tuples shown in Equation (2), including x, reasoning steps r, extracted

[1] Without these extensions, the standard Draft-Sketch-Prove paradigm Jiang et al. (2022) often yields incomplete mappings $x \to (r, \mathcal{H}(x), \phi(x))$, where implicit assumptions $h \notin \mathcal{H}(x)$ and inconsistencies in variable or unit definitions can lead to invalid or non-verifiable specifications $\phi(x)$ for which no valid proof trace $\pi \in \mathcal{P}_{valid}$ is obtained (see §6).

```
Instruction:  Given a scientific problem x, generate structured reasoning and a formal specification.

Example (Physics):                                    Formal Verification (Lean):

x: A perfect gas undergoes isothermal compression.    import Mathlib.Data.Real.Basic
Volume reduces by 2.20 dm^3. Final P = 5.04 bar,      import Mathlib.Tactic.NormNum
V = 4.65 dm^3. Find original pressure.

                                                      theorem prob_e1_2_a_a :
SLM(x) ->                                               (3.42 : ℝ) * (4.65 + 2.20) =
                                                            5.04 * 4.65 := by
{                                                       norm_num
  [DRAFT]  Boyle's Law:
           P1 * V1 = P2 * V2                          Verification:
           V1 = V2 + 2.20                             symbolically_verified

  [SKETCH] H(x) = {isothermal,                        Final Answer:
                   P2 = 5.04,                         3.42 bar
                   V2 = 4.65}

  [PROVE]  phi(x):
           P1 * (4.65 + 2.20) = 5.04 * 4.65
           \land 3.42 < P1 < 3.43
}
```

Figure 3: Physics prompt illustrating the mapping from x to $(r, \mathcal{H}(x), \phi(x))$ (Eq. 1), with formal verification encoding Boyle's-law relation and pressure; see §A for examples across other domains.

assumptions $\mathcal{H}(x)$, and encoded formal specification $\phi(x)$, along with either a Lean-verified proof $\pi \in \mathcal{P}_{valid}$ or an admissible counterexample x' satisfying $\neg h_x(x')$. The shorthand $\neg\phi(x')$ is used only when ϕ is defined as the corresponding claim predicate over the perturbed instance (see §A). Detailed dataset statistics are provided in §4.

$$\mathcal{D} = \{(x, r, \mathcal{H}(x), \phi(x), \pi \text{ or } x')\} \tag{2}$$

3.2 LEAN BENCHMARK

We define LEANBENCH as a benchmarking framework that evaluates five SLMs (1B-3B parameters) under verification-augmented scientific reasoning using LEANDATA. Within the neuro-symbolic loop, the model observes a proof state $s_t \in \mathcal{S}$, proposes a tactic $a_t \sim \pi_\theta(\cdot \mid s_t)$, and the proof assistant applies a deterministic transition $s_{t+1} = T(s_t, a_t)$. Execution feedback $\mathcal{F}_t = \{\text{validity}, \text{progress}, \text{error type}\}$ yields a step reward $r_t = R(\mathcal{F}_t)$ as defined in Equation (3), while the trajectory reward includes a terminal completion reward to distinguish partial proof progress from successful completion, with the policy optimized to maximize its expected value as defined in Equation (4), where $\beta > 0$ controls the terminal completion reward.

$$r_t = \begin{cases} +1 & \text{if } s_{t+1} \text{ advances the proof,} \\ 0 & \text{if no progress is made,} \\ -1 & \text{if the tactic is invalid or rejected} \end{cases} \tag{3}$$

$$R(\tau) = \sum_{t=1}^{T} r_t + \beta \mathbb{I}[s_T \text{ is a closed proof state}], \qquad \max_\theta \mathbb{E}_{\pi_\theta}[R(\tau)] \tag{4}$$

Given an input problem $x \in \mathcal{X}$, the trained SLM policies and baseline SLMs instantiate Equation (1) under zero-shot and few-shot prompting to generate structured outputs $(r, \mathcal{H}(x), \phi(x))$. For trained policies, $\phi(x)$ induces an initial proof state s_0, from which tactics $\pi = (a_1, \ldots, a_T)$ are generated via $\pi_\theta(a_t \mid s_t)$ under transitions $s_{t+1} = T(s_t, a_t)$. We evaluate models along two dimensions: (i) **Formal Verification**, which measures whether a valid proof trace $\pi \in \mathcal{P}_{valid}$ reaches a completed final state s_T; and (ii) **Counterexample-Based Falsification**, which measures whether the model identifies an admissible counterexample $x' \in \mathcal{X}$ satisfying $\neg h_x(x')$. For falsification instances, admissible candidate inputs x' are constructed by the controlled-variation procedure in §3.1, and the identified candidate is validated through formal execution. Counterexample-based falsification is applied only to instances whose formal specifications admit a valid falsifying input, while problems requiring a unique numerical solution are evaluated through formal verification. For the "w/o CEx" ablation, counterexample-generation supervision and training-time counterexample feedback are removed, while the same counterexample evaluation protocol is retained at test time. Evaluation is conducted within a closed neuro-symbolic loop in which $(r, \mathcal{H}(x), \phi(x))$ must satisfy structural consistency, alignment, and formal executability under T, assessing model outputs through valid Lean proof trajectories or their ability to expose violations through admissible counterexamples (see §6).

Table 1: Comparison of LEANDATA with prior datasets across reasoning r, hypotheses $\mathcal{H}(x)$, encoded formal specifications $\phi(x)$, Lean-verified proofs $\pi \in \mathcal{P}_{valid}$, and admissible counterexamples x' satisfying $\neg\phi(x')$.

Dataset	Size	Domain	r	$\mathcal{H}(x)$	$\phi(x)$	π	x'	$\lvert\mathbf{r}\rvert$	$\lvert\mathcal{H}(\mathbf{x})\rvert$	Verified
FIMO Liu et al. (2023)	149	Mathematics	✓	×	✓	✓	×	–	–	✓
INDIMATHBENCH Biyani et al. (2025)	312	Mathematics	✓	×	✓	✓	×	–	–	✓
PROOFNET Azerbayev et al. (2023)	371	Mathematics	✓	×	✓	✓	×	–	–	✓
MINIF2F Zheng et al. (2021)	488	Mathematics	✓	×	✓	✓	×	–	–	✓
SVAMP Patel et al. (2021)	1K	Arithmetic	✓	×	×	×	×	–	–	×
LEANDATA	580	Multi-Domain	✓	✓	✓	✓	✓	2.96	2.64	100%

4 EXPERIMENTAL SETUP

Dataset LEANDATA consists of 580 problems spanning physics (255), chemistry (154), probability (57), calculus (33), differential equations (30), statistics (22), thermodynamics (16), quantum mechanics (8), geometry (3), and linear algebra (2) (see §A). Each instance includes structured reasoning ($|r| = 2.96$ steps; $|\mathcal{H}(x)| = 2.64$ assumptions) and executable Lean 4 proofs, all symbolically verified (100%). As shown in Table 1, prior datasets provide only partial coverage, typically lacking alignment between r, $\mathcal{H}(x)$, and $\phi(x)$ or support for counterexample-based falsification. In contrast, LEANDATA unifies these components for formal verification ($\pi \in \mathcal{P}_{valid}$) and counterexample-based falsification ($\exists x' : \neg\phi(x')$) under formal validity constraints. Of the 580 problems, 244 (42.1%) support counterexample-based falsification and 336 (57.9%) formal verification.

Evaluation Metrics We evaluate models under verification-augmented scientific reasoning using five metrics, such as Verification Accuracy (VA), Falsification Rate (FR), Structural Consistency (SC), Alignment Consistency (AC), and End-to-End (E2E) success. Let $\mathcal{I}_{\text{proof}}$ and $\mathcal{I}_{\text{cex}}$ denote instances requiring formal verification and admitting counterexample-based falsification, respectively, with $|\mathcal{I}_{\text{proof}}| = 336$, $|\mathcal{I}_{\text{cex}}| = 244$, and $\mathcal{I}_{\text{proof}} \cap \mathcal{I}_{\text{cex}} = \emptyset$. VA is $\text{VA} = \frac{1}{|\mathcal{I}_{\text{proof}}|} \sum_{i \in \mathcal{I}_{\text{proof}}} \mathbb{I}[\pi_i \in \mathcal{P}_{\text{valid}}]$ for formally valid proofs; FR is $\text{FR} = \frac{1}{|\mathcal{I}_{\text{cex}}|} \sum_{i \in \mathcal{I}_{\text{cex}}} \mathbb{I}[\exists x'_i \in \mathcal{X} : \neg\phi_i(x'_i)]$ for counterexample identification; SC is $\text{SC} = \frac{1}{N} \sum_{i=1}^{N} \mathbb{I}[(r_i, \mathcal{H}(x_i), \phi(x_i))$ are well-formed$]$ for valid variables and expressions; AC is $\text{AC} = \frac{1}{N} \sum_{i=1}^{N} \mathbb{I}[r_i \leftrightarrow \mathcal{H}(x_i) \leftrightarrow \phi(x_i)]$ for cross-stage agreement; and E2E is $\text{E2E} = \frac{1}{N} \sum_{i=1}^{N} \mathbb{I}[\text{SC}_i \wedge \text{AC}_i \wedge (\mathbb{I}[i \in \mathcal{I}_{\text{proof}}]\mathbb{I}[\pi_i \in \mathcal{P}_{\text{valid}}] \vee \mathbb{I}[i \in \mathcal{I}_{\text{cex}}]\mathbb{I}[\exists x'_i \in \mathcal{X} : \neg\phi_i(x'_i)])]$ for successful end-to-end formal verification or counterexample-based falsification subject to structural and alignment consistency. All metrics are reported as percentages, with higher values indicating better performance.

Hyperparameters We train an SLM to learn $\pi_\theta(a_t \mid s_t)$ within the neuro-symbolic loop in Eqs (3)-(4) using reinforcement learning with $T = 32$ and $\gamma = 1.0$. We use a learning rate of 3×10^{-5} with Adam ($\beta_1 = 0.9$, $\beta_2 = 0.95$) for 3 epochs over LEANDATA, accumulating gradients over 8 steps with per-device batch size 4 (effective batch size 32). We apply gradient clipping (norm 1.0), temperature sampling ($\tau = 0.7$), and $K = 3$ maximum attempts for re-prompting and counterexample generation. Experiments use a single NVIDIA A100 GPU (80GB), with peak training memory of approximately 40GB-48GB. Sensitivity of T, τ, and K is analyzed in §6.

Baselines We evaluate five SLM backbones (1B-3B): Phi-2[2], TinyLlama[3], Qwen2-1.5B[4], Gemma-2B[5], and DeepSeek-1.3B[6], under zero-shot and few-shot prompting. Each backbone is evaluated under *Base*, *+Supervised*, and *+RL+Verification*, generating $(r, \mathcal{H}(x), \phi(x))$ following Eq. (1) and evaluated on **Formal Verification** and **Counterexample-Based Falsification**. *Base* uses no additional training; *+Supervised* applies structured supervision on LEANDATA without RL or verification-guided optimization; and *+RL+Verification* optimizes $\pi_\theta(a_t \mid s_t)$ via RL with proof-assistant execution feedback and verification-guided rewards. For **Formal Verification**, Base and +Supervised submit $\phi(x)$ to Lean, whereas +RL+Verification additionally generates tactics via $\pi_\theta(a_t \mid s_t)$. For **Counterexample-Based Falsification**, all conditions are evaluated on falsifiable instances and must identify an admissible $x' \in \mathcal{X}$ satisfying $\neg h_x(x')$. All trained conditions use the same LEANDATA splits and formal evaluation settings.

[2] https://huggingface.co/microsoft/phi-2

[3] https://huggingface.co/TinyLlama/TinyLlama-1.1B-Chat-v1.0

[4] https://huggingface.co/Qwen/Qwen2-1.5B

[5] https://huggingface.co/google/gemma-2b

[6] https://huggingface.co/deepseek-ai/deepseek-coder-1.3b-instruct

Table 2: Main results on LEANDATA. We evaluate SLMs (1B-3B) under zero-shot (ZS) and few-shot (FS) prompting across Base (prompt-only), +Supervised (same-data structured supervision), and +RL+Verification (full method). Compute includes throughput (Tok/s ↑) and memory (GB ↓).

	Zero-shot (ZS)						Few-shot (FS)						Compute	
Model	VA	FR	SC	AC	E2E	Avg	VA	FR	SC	AC	E2E	Avg	Tok/s	Mem
Base (Prompt-Only Baselines)														
Phi-2	21.4	18.2	62.5	48.1	29.6	36.0	27.8	22.6	68.3	53.7	36.9	41.9	165	18
TinyLlama	15.2	12.4	55.1	41.3	21.7	29.1	19.6	15.8	60.7	45.9	26.8	33.8	**210**	**12**
Qwen2-1.5B	23.7	20.1	64.2	50.5	32.8	38.3	29.9	24.3	70.4	56.8	39.5	44.2	175	16
Gemma-2B	**25.6**	**21.7**	**66.8**	**52.9**	**34.7**	**40.3**	**32.4**	**26.8**	**72.9**	**58.6**	**41.9**	**46.5**	150	20
DeepSeek-1.3B	24.1	19.5	63.7	49.8	33.2	38.1	30.8	24.9	69.5	55.4	40.3	44.2	180	15
+Supervised (Same-Data Structured Supervision)														
Phi-2 +Supervised	24.9	21.3	66.8	52.6	34.8	40.1	31.5	26.2	72.1	58.4	42.3	46.1	160	22
TinyLlama +Supervised	18.7	15.4	59.8	45.6	27.5	33.4	23.8	19.6	65.9	50.2	32.1	38.3	**205**	**14**
Qwen2-1.5B +Supervised	27.9	23.8	69.5	55.1	38.6	43.0	34.6	28.5	74.8	60.7	45.9	48.9	170	18
Gemma-2B +Supervised	**29.8**	**25.6**	**71.4**	**57.9**	**40.7**	**45.1**	**37.2**	**30.4**	**77.3**	**63.1**	**48.6**	**51.3**	145	24
DeepSeek-1.3B +Supervised	28.6	24.2	70.2	56.3	39.5	43.8	35.9	29.1	75.6	61.4	47.2	49.8	175	17
+RL + Verification (Full Method)														
Phi-2 +RL+Verif	39.6	33.2	76.8	69.1	49.7	53.7	44.8	38.1	80.2	73.5	55.6	58.4	155	26
TinyLlama +RL+Verif	31.2	26.4	70.1	62.7	40.2	46.1	36.7	30.9	74.5	67.8	47.3	51.4	**200**	**16**
Qwen2-1.5B +RL+Verif	41.8	35.6	78.9	71.8	52.9	56.2	46.9	39.7	81.6	75.1	57.8	60.2	165	22
Gemma-2B +RL+Verif	**43.9**	**37.5**	**80.3**	**73.2**	**55.1**	**58.0**	**48.6**	**41.2**	**82.7**	**76.4**	**58.9**	**61.6**	140	28
DeepSeek-1.3B +RL+Verif	42.7	36.4	79.8	72.6	54.3	57.2	47.3	40.5	82.1	75.8	58.2	60.8	170	23

Table 3: Domain-wise performance on LEANDATA using Gemma-2B, comparing +Trained and +RL+Verification under zero-shot (ZS) and few-shot (FS) prompting.

	Zero-shot (ZS)						Few-shot (FS)						Compute	
Domain	VA	FR	SC	AC	E2E	Avg	VA	FR	SC	AC	E2E	Avg	Tok/s	Mem
+Trained (Ours w/o Verification RL)														
Physics	30.0	25.7	71.8	58.2	41.0	45.3	37.5	30.8	77.4	63.6	49.6	51.8	**146**	22.3
Chemistry	28.7	24.5	70.4	56.8	39.7	44.0	35.8	28.7	76.2	62.1	46.2	49.8	139	21.7
Probability	29.4	25.2	71.1	57.5	40.3	44.7	36.5	29.8	76.9	62.7	47.0	50.6	145	23.1
Calculus	33.0	28.2	**74.5**	60.7	44.8	48.2	40.6	33.1	80.1	65.9	53.2	54.6	142	23.4
Differential Equations	31.2	26.7	73.1	59.3	42.3	46.5	38.4	31.5	78.5	64.5	48.9	52.4	138	**21.2**
Statistics	29.1	24.9	71.0	57.2	40.0	44.4	36.2	29.4	76.6	62.3	46.6	50.2	144	23.6
Thermodynamics	28.5	24.3	70.2	56.6	39.3	43.8	35.5	28.6	76.0	61.7	45.8	49.5	140	22.1
Quantum Mechanics	27.9	23.8	69.6	56.1	38.8	43.2	34.9	28.1	75.4	61.2	45.1	48.9	143	21.9
Geometry	33.3	33.3	66.7	**66.7**	33.3	46.7	**66.7**	**66.7**	**100.0**	**66.7**	66.7	**73.3**	141	22.8
Linear Algebra	**50.0**	**50.0**	50.0	50.0	**50.0**	**50.0**	50.0	50.0	100.0	50.0	**100.0**	70.0	137	22.6
+RL + Verification (Full Method)														
Physics	44.0	37.4	80.4	73.5	55.2	58.1	48.9	41.7	82.9	77.0	59.3	62.0	137.8	28.3
Chemistry	42.5	36.2	79.1	72.2	53.8	56.8	46.6	39.6	81.5	75.2	56.8	59.9	136.2	**27.5**
Probability	43.4	37.0	79.8	73.1	54.5	57.6	47.8	40.5	82.3	76.4	58.1	61.0	139.4	28.9
Calculus	48.3	41.7	83.9	**77.6**	60.6	62.4	51.8	43.9	84.8	**79.3**	62.4	64.4	137.1	27.6
Differential Equations	45.3	38.8	81.7	75.3	56.7	59.6	49.7	42.3	83.7	77.9	60.2	62.8	135.6	27.8
Statistics	43.1	36.6	79.5	72.6	54.2	57.2	47.1	40.1	82.0	75.8	57.5	60.5	138.3	27.9
Thermodynamics	42.4	35.9	78.9	71.9	53.5	56.5	46.2	39.2	81.2	74.9	56.4	59.6	138.7	28.1
Quantum Mechanics	41.6	35.2	78.2	71.3	52.8	55.8	45.3	38.5	80.6	74.3	55.6	58.9	**140.1**	29.2
Geometry	**66.7**	**66.7**	**100.0**	66.7	**66.7**	**73.3**	**100.0**	**66.7**	**100.0**	66.7	**100.0**	**86.7**	135.6	27.8
Linear Algebra	50.0	50.0	100.0	50.0	50.0	60.0	100.0	50.0	100.0	50.0	100.0	80.0	136.8	28.7

5 RESULTS AND ANALYSIS

Benchmark Analysis. Table 2 shows that LEANDATA remains challenging due to the need for unified consistent reasoning r, assumptions $\mathcal{H}(x)$, and executable specifications $\phi(x)$. Prompt-only baselines achieve limited performance under zero-and few-shot prompting (VA $\leq 32.4\%$, E2E $\leq 41.9\%$), with higher SC/AC not consistently translating to formal proof validity $\pi \in \mathcal{P}_{valid}$ or counterexample-based falsification. To control for dataset effects, +Supervised models use the same structured outputs $(r, \mathcal{H}(x), \phi(x))$, proof environment, and evaluation protocol without RL or verification feedback; structured supervision improves performance (Gemma-2B E2E $41.9\% \rightarrow 48.6\%$) but remains below +RL+Verification. The latter improves all five metrics across SLMs; for Gemma-2B, few-shot VA, FR, SC, AC, and E2E increase from $32.4\%, 26.8\%, 72.9\%, 58.6\%, 41.9\%$ to $48.6\%, 41.2\%, 82.7\%, 76.4\%, 58.9\%$, respectively. Under zero-shot prompting, VA increases $25.6\% \rightarrow 43.9\%$, E2E $34.7\% \rightarrow 55.1\%$, and Avg $40.3\% \rightarrow 58.0\%$. These gains incur lower throughput and higher memory usage, reflecting the compute cost of verification-guided optimization.

Domain-Wise Analysis. Table 3 presents domain-wise results for Gemma-2B, highlighting the heterogeneous difficulty of LEANDATA. +RL+Verification improves E2E across nine domains, with

Table 4: Ablation analysis on LEANDATA using Gemma-2B, evaluating key components under zero-shot (ZS) and few-shot (FS) prompting. Abbreviations: DSP = Draft-Sketch-Prove Extensions; RL = Reinforcement Learning; Ver. = Formal Verification; Struct. = Structural Checks; Align. = Alignment Checks; CEx. = Counterexample Generation; CL = Closed-Loop Execution.

	Zero-shot (ZS)						Few-shot (FS)						Compute	
Setting	VA	FR	SC	AC	E2E	Avg	VA	FR	SC	AC	E2E	Avg	Tok/s	Mem
Base	25.6	21.7	66.8	52.9	34.7	40.3	32.4	26.8	72.9	58.6	41.9	46.5	147.8	**21.4**
+ Super. (w/ DSP, w/o RL, w/o Ver.)	29.8	25.6	71.4	57.9	40.7	45.1	37.2	30.4	77.3	63.1	48.6	51.3	140.4	23.5
+ RL (w/ DSP, w/ RL, w/o Ver.)	32.9	28.4	73.8	61.5	43.6	48.0	40.1	32.7	78.5	66.4	51.2	53.8	143.6	27.8
+ Ver. (w/ DSP, w/o RL, w/ Ver.)	34.2	30.1	75.1	63.8	45.3	49.7	41.3	35.6	79.6	68.9	53.4	55.8	**151.3**	24.1
+ RL+Ver. (w/ DSP, w/ RL, w/ Ver.)	**43.9**	**37.5**	**80.3**	**73.2**	**55.1**	**58.0**	**48.6**	**41.2**	**82.7**	**76.4**	**58.9**	**61.6**	145.2	24.9
w/o DSP (w/ RL, w/ Ver.)	31.5	27.2	70.3	59.8	42.6	46.3	38.4	31.0	75.1	64.2	49.1	51.6	143.2	24.8
w/o Struct. (w/ DSP, w/ RL, w/ Ver.)	33.1	29.4	68.2	60.5	43.8	47.0	40.2	33.8	72.4	65.1	50.3	52.4	142.7	**21.1**
w/o Align. (w/ DSP, w/ RL, w/ Ver.)	34.0	30.8	74.1	61.2	44.9	49.0	41.5	35.2	78.3	66.0	52.1	54.6	141.9	26.2
w/o CEx. (w/ DSP, w/ RL, w/ Ver.)	**38.7**	24.1	**79.5**	**72.4**	**49.3**	**52.8**	**45.1**	28.6	**82.1**	**75.3**	**54.7**	57.2	142.3	25.6
w/o CL (w/ DSP, w/ RL, w/ Ver.)	36.2	**31.5**	77.6	69.8	47.6	52.5	43.3	**36.9**	81.2	73.1	53.2	**57.5**	**144.1**	25.1

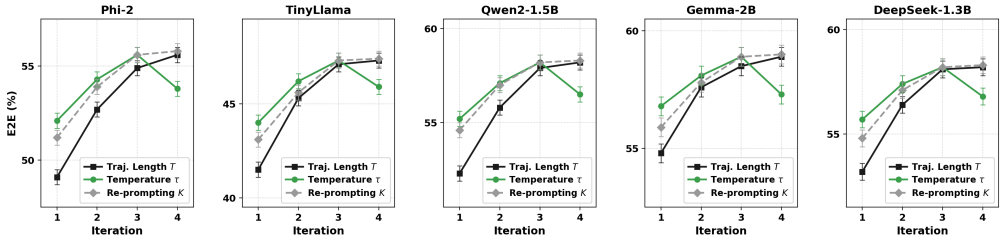


Figure 4: Hyperparameter sensitivity on LEANDATA, varying trajectory length T, temperature τ, and re-prompting attempts K. Results report E2E (%) under few-shot (FS) across five SLMs, measuring success via valid proofs ($\pi \in \mathcal{P}_{valid}$) or admissible counterexamples ($\exists x' : \neg\phi(x')$).

Linear Algebra unchanged, and yields consistent gains in VA, FR, SC, and AC across the seven larger domains. For example, physics improves from $41.0\% \rightarrow 55.2\%$ and $49.6\% \rightarrow 59.3\%$ under zero-shot and few-shot prompting, while calculus improves from $44.8\% \rightarrow 60.6\%$ and $53.2\% \rightarrow 62.4\%$. Across the seven larger domains, SC increases from 70.2%-74.5% to 78.9%-83.9% (ZS) and from 76.0%-80.1% to 81.2%-84.8% (FS), with similar gains in AC. Geometry and Linear Algebra contain only 3 and 2 instances, respectively, making their reported percentages highly sensitive to individual examples and therefore less representative. +RL+Verification also reduces throughput ($\sim$136-140 Tok/s) and increases memory ($\sim$28-29 GB). Results for the remaining four models are in Appendix A.

6 ABLATION STUDY

Table 4 presents an ablation analysis under zero-shot and few-shot prompting. Structured supervision improves E2E from $34.7\% \rightarrow 40.7\%$ (ZS) and $41.9\% \rightarrow 48.6\%$ (FS), while RL further improves it to 43.6% and 51.2%, formal verification alone to 45.3% and 53.4%, and the full method to 55.1% and 58.9%, with VA of $43.9\%/48.6\%$ and FR of $37.5\%/41.2\%$ (ZS/FS). Removing DSP, structural checks, or alignment checks reduces E2E to $42.6\%/49.1\%$, $43.8\%/50.3\%$, and $44.9\%/52.1\%$, respectively, while removing counterexample generation reduces FR to $24.1\%/28.6\%$ and closed-loop execution reduces VA to $36.2\%/43.3\%$ and E2E to $47.6\%/53.2\%$. The full method incurs moderate computational overhead (24.9 GB memory) with comparable inference throughput. Detailed formal verification analysis is provided in Appendix A, with results for the remaining four models in Appendix A.

Hyperparameter Sensitivity and Error Analysis. We analyze key hyperparameters and errors across all five SLMs (see Figure 4 and Figure 5). Increasing T improves performance with gains saturating near $T = 32$, while $\tau = 0.7$ balances exploration and stability; lower values reduce diversity, whereas higher values introduce noisy or invalid reasoning steps. Increasing K improves performance through iterative refinement and recovery from invalid generations, with gains stabilizing at $K = 3$. Formal verification yields moderate success rates (36.7%-48.6%), while human evaluation shows higher semantic correctness (60.2%-71.2% correct; 18.5%-24.3% partially correct), indicating that some formally invalid outputs still capture meaningful reasoning (see §6). This gap suggests failures can arise from invalid formal specifications, proof-assistant execution, or misalignment between generated components and formal constraints.

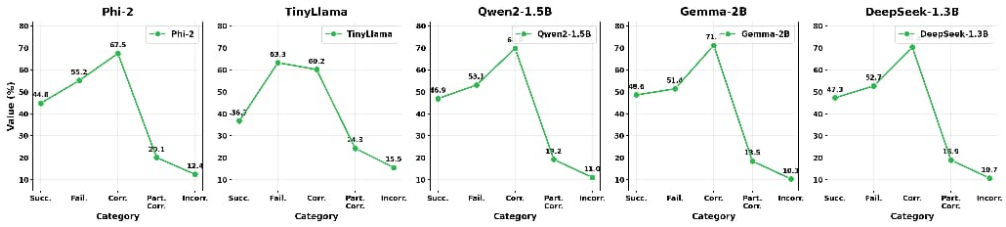


Figure 5: Error analysis on LEANDATA across five SLMs under +RL+Verification. Formal verification is binary ($\pi \in \mathcal{P}_{valid}$ or not), while human evaluation assesses semantic correctness on a sampled subset. All values are in %.

Table 5: Human evaluation on LEANDATA under zero-shot (ZS) and few-shot (FS) settings. The table reports semantic correctness, annotation consistency, and faithfulness of $(r, \mathcal{H}(x), \phi(x))$ on 200 randomly sampled problems across ten scientific domains. See definitions in A.

Evaluation	Criterion	ZS Avg	FS Avg	Agreement	κ	Consistency	Var
Human	Informal Annotation	46.8	51.2	0.71	0.64	0.69	7.8
Human	Small-Sample Annotation	48.5	53.1	0.74	0.67	0.72	6.9
Human	Rubric-Guided Annotation	53.7	58.4	0.86	0.81	0.84	3.4
Human	Rubric + Cross-Verification	**56.2**	**60.8**	**0.91**	**0.87**	**0.89**	**2.6**
Faithfulness	**Criterion**	**Score**	**Full**	**Partial**	**Missing**		
Faithfulness	Equation Fidelity	96.5	96.5	3.0	0.5		
Faithfulness	Constraint Fidelity	94.8	98.0	1.5	0.5		
Faithfulness	Variable & Unit Consistency	97.2	97.5	2.0	0.5		
Faithfulness	Semantic Fidelity	95.4	93.5	5.0	1.5		
Faithfulness	Lean Executability	100.0	99.0	1.0	0.0		
Faithfulness	**Overall Faithfulness**	**96.78**	**91.5**	**7.0**	**1.5**		
IAA	$N = 200$	**0.91**	183	14	3		

Table 6: Generalization analysis of LEANDATA under zero-shot (ZS) and few-shot (FS) evaluation, reporting Avg performance across a random 80/10/10 instance-level split into train (80%), validation (10%), and test (10%) sets for each model.

+RL+Verification Models							
Model	**Split Type**	**Zero-Shot (ZS) Avg**			**Few-Shot (FS) Avg**		
		Train	**Val**	**Test**	**Train**	**Val**	**Test**
Phi-2 +RL+Verif	Random instance split (80/10/10)	58.1	55.4	53.7	63.2	60.1	58.4
TinyLlama +RL+Verif	Random instance split (80/10/10)	49.6	47.8	46.1	55.3	53.1	51.4
Qwen2-1.5B +RL+Verif	Random instance split (80/10/10)	60.7	58.4	56.2	64.9	62.3	60.2
Gemma-2B +RL+Verif	Random instance split (80/10/10)	62.5	60.2	58.0	66.1	63.8	61.6
DeepSeek-1.3B +RL+Verif	Random instance split (80/10/10)	**61.8**	**59.5**	**57.2**	**65.4**	**63.1**	**60.8**

Human Evaluation Analysis. We compare unrestricted, sample-controlled, rubric-guided, and cross-verified settings (see Table 5). Informal annotation yields lower agreement and higher variance (Agreement: 0.71; κ: 0.64; Var: 7.8), while rubric-guided evaluation improves consistency (Agreement: 0.86-0.91; κ: 0.81-0.87; Var: 2.6-3.4). Across 200 randomly sampled problems spanning all ten domains, Overall Faithfulness reaches 96.78%, with 91.5% (183/200) fully encoding all required components and IAA of 0.91.

Generalization Analysis. We evaluate all models under a random 80/10/10 instance-level split into training, validation, and test sets under zero-shot and few-shot prompting to assess generalization beyond training instances (see Table 6). Across SLM backbones, Avg scores decrease from training to validation and test; for Gemma-2B +RL+Verif, they decrease from $62.5\% \rightarrow 58.0\%$ (ZS) and $66.1\% \rightarrow 61.6\%$ (FS). Similar trends with relatively small validation-test gaps indicate stable generalization beyond training instances and that gains are not restricted to same-instance evaluation or training-data memorization.

7 CONCLUSION

We show that verification-augmented reasoning improves SLMs' scientific reasoning by including formal validity and supporting counterexample-based falsification, with consistent gains across domains. Despite these gains, the gap between formal validity and semantic correctness highlights limitations in specification alignment and proof-assistant execution. Future work will extend the work to broader scientific domains and strengthen the integration of formal verification with natural-language reasoning.

AI USE STATEMENT

In this work, we used generative AI tools to aid and polish manuscript writing and to generate synthetic datasets. We did not use generative AI tools to generate or modify experimental results or scientific claims; the remaining required disclosure tasks are not applicable to this work. We reviewed all AI-assisted content and verified the final manuscript. We take responsibility for the final content of this work, including text, claims, and artifacts produced with the aid of generative AI.

ETHICS STATEMENT

The work does not involve crowdsourcing, sensitive personal data, or high-risk deployment scenarios. Human evaluation was conducted without compensation by male members of the same research group, aged 25-30, to assess model outputs within a controlled Lean-based formal reasoning environment. The released LEANDATA dataset consists of JSON files, and we follow the licenses and terms of use of the datasets and other prior assets used in this work.

REPRODUCIBILITY STATEMENT

The work provides the information required to reproduce LEANDATA construction and validation in §3.1, §3.2, and §4. The LEANDATA dataset is provided as a JSON file in the supplementary materials and contains 580 instances across 10 domains, with structured fields for problems, domains, difficulty, reasoning steps, equations, hypotheses, Lean formal statements and proof scripts, verification results, final answers, source datasets, and units.

REFERENCES

Abdelrahman Abdallah, Bhawna Piryani, Jamshid Mozafari, Mohammed Ali, and Adam Jatowt. How good are llm-based rerankers? an empirical analysis of state-of-the-art reranking models. *arXiv preprint arXiv:2508.16757*, 2025.

Akari Asai, Zeqiu Wu, Yizhong Wang, Avi Sil, and Hannaneh Hajishirzi. Self-rag: Learning to retrieve, generate, and critique through self-reflection. In *International conference on learning representations*, volume 2024, pp. 9112–9141, 2024.

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.

Peter Belcak, Greg Heinrich, Shizhe Diao, Yonggan Fu, Xin Dong, Saurav Muralidharan, Yingyan Celine Lin, and Pavlo Molchanov. Small language models are the future of agentic ai. *arXiv preprint arXiv:2506.02153*, 2025.

Yves Bertot and Pierre Castéran. *Interactive theorem proving and program development: Coq'Art: the calculus of inductive constructions*. Springer Science & Business Media, 2013.

Param Biyani, Shashank Kirtania, Yasharth Bajpai, Sumit Gulwani, and Ashish Tiwari. Indimathbench: Autoformalizing mathematical reasoning problems with a human touch. *arXiv preprint arXiv:2512.00997*, 2025.

Wenhu Chen, Xueguang Ma, Xinyi Wang, and William W Cohen. Program of thoughts prompting: Disentangling computation from reasoning for numerical reasoning tasks. *arXiv preprint arXiv:2211.12588*, 2022.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, et al. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*, 2021.

Yuxian Gu, Li Dong, Furu Wei, and Minlie Huang. Minillm: Knowledge distillation of large language models. In *International Conference on Learning Representations*, volume 2024, pp. 32694–32717, 2024.

Thomas Hubert, Rishi Mehta, Laurent Sartran, Miklós Z Horváth, Goran Žužić, Eric Wieser, Aja Huang, Julian Schrittwieser, Yannick Schroecker, Hussain Masoom, et al. Olympiad-level formal mathematical reasoning with reinforcement learning. *Nature*, 651(8106):607–613, 2026.

Mojan Javaheripi, Sébastien Bubeck, Marah Abdin, Jyoti Aneja, Sebastien Bubeck, Caio César Teodoro Mendes, Weizhu Chen, Allie Del Giorno, Ronen Eldan, Sivakanth Gopi, et al. Phi-2: The surprising power of small language models. *Microsoft Research Blog*, 1(3):3, 2023.

Albert Q Jiang, Sean Welleck, Jin Peng Zhou, Wenda Li, Jiacheng Liu, Mateja Jamnik, Timothée Lacroix, Yuhuai Wu, and Guillaume Lample. Draft, sketch, and prove: Guiding formal theorem provers with informal proofs. *arXiv preprint arXiv:2210.12283*, 2022.

Keigo Kusumegi, Xinyu Yang, Paul Ginsparg, Mathijs De Vaan, Toby Stuart, and Yian Yin. Scientific production in the era of large language models. *Science*, 390(6779):1240–1243, 2025.

Yinghui Li, Jiayi Kuang, Haojing Huang, Zhikun Xu, Xinnian Liang, Yi Yu, Wenlian Lu, Yangning Li, Xiaoyu Tan, Chao Qu, et al. One example shown, many concepts known! counterexample-driven conceptual reasoning in mathematical llms. *arXiv preprint arXiv:2502.10454*, 2025.

Zenan Li, Zhaoyu Li, Kaiyu Yang, Xiaoxing Ma, and Zhendong Su. Learning to disprove: Formal counterexample generation with large language models. *arXiv preprint arXiv:2603.19514*, 2026.

Hunter Lightman, Vineet Kosaraju, Yuri Burda, Harrison Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. Let's verify step by step. In *International Conference on Learning Representations*, volume 2024, pp. 39578–39601, 2024.

Yu-Chen Lin, Sanat Sharma, Hari Manikandan, Jayant Kumar, Tracy Holloway King, and Jing Zheng. Efficient multitask learning in small language models through upside-down reinforcement learning. *arXiv preprint arXiv:2502.09854*, 2025.

Chengwu Liu, Jianhao Shen, Huajian Xin, Zhengying Liu, Ye Yuan, Haiming Wang, Wei Ju, Chuanyang Zheng, Yichun Yin, Lin Li, et al. Fimo: A challenge formal dataset for automated theorem proving. *arXiv preprint arXiv:2309.04295*, 2023.

Aman Madaan, Niket Tandon, Prakhar Gupta, Skyler Hallinan, Luyu Gao, Sarah Wiegreffe, Uri Alon, Nouha Dziri, Shrimai Prabhumoye, Yiming Yang, et al. Self-refine: Iterative refinement with self-feedback. *Advances in neural information processing systems*, 36:46534–46594, 2023.

Leonardo de Moura and Sebastian Ullrich. The lean 4 theorem prover and programming language. In *International Conference on Automated Deduction*, pp. 625–635. Springer, 2021.

Tobias Nipkow, Markus Wenzel, and Lawrence C Paulson. *Isabelle/HOL: a proof assistant for higher-order logic*. Springer, 2002.

Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, et al. Training language models to follow instructions with human feedback. *Advances in neural information processing systems*, 35: 27730–27744, 2022.

Arkil Patel, Satwik Bhattamishra, and Navin Goyal. Are nlp models really able to solve simple math word problems? In *Proceedings of the 2021 conference of the North American chapter of the association for computational linguistics: human language technologies*, pp. 2080–2094, 2021.

Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter. *arXiv preprint arXiv:1910.01108*, 2019.

Timo Schick, Jane Dwivedi-Yu, Roberto Dessì, Roberta Raileanu, Maria Lomeli, Eric Hambro, Luke Zettlemoyer, Nicola Cancedda, and Thomas Scialom. Toolformer: Language models can teach themselves to use tools. *Advances in neural information processing systems*, 36:68539–68551, 2023.

Raghav Sharma and Manan Mehta. Small language models for agentic systems: A survey of architectures, capabilities, and deployment trade offs. *arXiv preprint arXiv:2510.03847*, 2025.

Peiyang Song, Kaiyu Yang, and Anima Anandkumar. Lean copilot: Large language models as copilots for theorem proving in lean. *arXiv preprint arXiv:2404.12534*, 2024.

Zhangde Song, Jieyu Lu, Yuanqi Du, Botao Yu, Thomas M Pruyn, Yue Huang, Kehan Guo, Xiuzhe Luo, Yuanhao Qu, Yi Qu, et al. Evaluating large language models in scientific discovery. *arXiv preprint arXiv:2512.15567*, 2025.

Yang Sui, Yu-Neng Chuang, Guanchu Wang, Jiamu Zhang, Tianyi Zhang, Jiayi Yuan, Hongyi Liu, Andrew Wen, Shaochen Zhong, Na Zou, et al. Stop overthinking: A survey on efficient reasoning for large language models. *arXiv preprint arXiv:2503.16419*, 2025.

Karthik Valmeekam, Alberto Olmo, Sarath Sreedharan, and Subbarao Kambhampati. Large language models still can't plan (a benchmark for llms on planning and reasoning about change). In *NeurIPS 2022 Foundation Models for Decision Making Workshop*, 2022.

Jiacheng Wang, Yejun Zeng, Jinyang Guo, Yuqing Ma, Aishan Liu, and Xianglong Liu. Slmquant: Benchmarking small language model quantization for practical deployment. In *Proceedings of the 3rd International Workshop on Rich Media With Generative AI*, pp. 2–10, 2025.

Xiaoxuan Wang, Ziniu Hu, Pan Lu, Yanqiao Zhu, Jieyu Zhang, Satyen Subramaniam, Arjun R Loomba, Shichang Zhang, Yizhou Sun, and Wei Wang. Scibench: Evaluating college-level scientific problem-solving abilities of large language models. *arXiv preprint arXiv:2307.10635*, 2023.

Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc Le, Ed Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. Self-consistency improves chain of thought reasoning in language models. *arXiv preprint arXiv:2203.11171*, 2022.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, et al. Chain-of-thought prompting elicits reasoning in large language models. *Advances in neural information processing systems*, 35:24824–24837, 2022.

Huajian Xin, ZZ Ren, Junxiao Song, Zhihong Shao, Wanjia Zhao, Haocheng Wang, Bo Liu, Liyue Zhang, Xuan Lu, Qiushi Du, et al. Deepseek-prover-v1. 5: Harnessing proof assistant feedback for reinforcement learning and monte-carlo tree search. In *International Conference on Learning Representations*, volume 2025, pp. 72274–72303, 2025.

Kaiyu Yang, Aidan Swope, Alex Gu, Rahul Chalamala, Peiyang Song, Shixing Yu, Saad Godil, Ryan J Prenger, and Animashree Anandkumar. Leandojo: Theorem proving with retrieval-augmented language models. *Advances in Neural Information Processing Systems*, 36:21573–21612, 2023.

Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Tom Griffiths, Yuan Cao, and Karthik Narasimhan. Tree of thoughts: Deliberate problem solving with large language models. *Advances in neural information processing systems*, 36:11809–11822, 2023.

Peiyuan Zhang, Guangtao Zeng, Tianduo Wang, and Wei Lu. Tinyllama: An open-source small language model. *arXiv preprint arXiv:2401.02385*, 2024.

Yanbo Zhang, Sumeer A Khan, Adnan Mahmud, Huck Yang, Alexander Lavin, Michael Levin, Jeremy Frey, Jared Dunnmon, James Evans, Alan Bundy, et al. Advancing the scientific method with large language models: From hypothesis to discovery. *arXiv preprint arXiv:2505.16477*, 2025.

Kunhao Zheng, Jesse Michael Han, and Stanislas Polu. Minif2f: a cross-system benchmark for formal olympiad-level mathematics. *arXiv preprint arXiv:2109.00110*, 2021.

A Appendix

Multi-Domain Prompt Analysis. Figures 6-14 illustrate representative examples across nine scientific domains, demonstrating the generality of the Draft-Sketch-Prove mapping, where the SLM decomposes problems into structured reasoning r, extracts domain-specific assumptions $\mathcal{H}(x)$, and

produces encoded formal specifications $\phi(x)$ executable in Lean for structural, alignment, and formal verification checks. Formal verification establishes formal validity when the encoded specification captures the target relation, including the relevant equations, constraints, assumptions, and final claim, and admits a valid Lean proof trace $\pi \in \mathcal{P}_{valid}$. For falsification instances, the same encoded formal specification supports counterexample-based falsification by identifying admissible inputs x' that violate the encoded hypothesis, providing a unified representation for formal verification and counterexample-based falsification.

```
Instruction:  Given a scientific problem x, generate structured reasoning and a formal specification.

Example (Chemistry):                                  Formal Verification (Lean):

x: 255 mg neon occupies                               import Mathlib.Data.Real.Basic
3.00 dm^3 at 122 K.                                   import Mathlib.Tactic.NormNum
Find pressure using
perfect gas law.                                      theorem prob_e1_4_a :
                                                        ((0.255 / 20.18) * 8.314 * 122) / 3.00 =
SLM(x) ->                                                   (0.042 : ℝ) := by
                                                        norm_num
{
  [DRAFT]  PV = nRT                                   Verification:
           n = m / M                                  symbolically_verified
           Convert units
                                                      Final Answer:
  [SKETCH] H(x) = {m = 255 mg,                        0.042 bar
                   V = 3.00,
                   T = 122,
                   R = 8.314,
                   M = 20.18}

  [PROVE]  phi(x):
           P = (m/M)RT/V
           \land 0.042 < P < 0.043
}
```

Figure 6: Prompt examples of chemistry illustrating stage-specific instructions for mapping an input problem x to $(r, \mathcal{H}(x), \phi(x))$ via a 1B-3B-parameter SLM, consistent with Equation (1).

```
Instruction:  Given a scientific problem x, generate structured reasoning and a formal specification.

Example (Probability):                                Formal Verification (Lean):

x: Two coins are tossed.                              import Mathlib.Data.Real.Basic
Let w = number of heads.                              import Mathlib.Tactic.NormNum
Find expected value <w>.
                                                      theorem prob_5_8 :
SLM(x) ->                                               (0 : ℝ) * (1/4) +
                                                        1 * (1/2) +
{                                                       2 * (1/4) = 1 := by
  [DRAFT]  E[w] = Sum w*P(w)                            norm_num
           = 0*1/4 + 1*1/2 + 2*1/4
                                                      Verification:
  [SKETCH] H(x) = {P(0)=1/4,                          symbolically_verified
                   P(1)=1/2,
                   P(2)=1/4}                          Final Answer:
                                                      1
  [PROVE]  phi(x):
           0*(1/4) + 1*(1/2) + 2*(1/4) = 1
}
```

Figure 7: Prompt examples of probability illustrating stage-specific instructions for mapping an input problem x to $(r, \mathcal{H}(x), \phi(x))$ via a 1B-3B-parameter SLM, consistent with Equation (1).

```
Instruction:  Given a scientific problem x, generate structured reasoning and a formal specification.

Example (Calculus):                                  Formal Verification (Lean):

x: Evaluate ∫0^{0.4}                                 import Mathlib.Data.Real.Basic
Γ(7)/(Γ(4)Γ(3)) y^3(1-y)^2 dy.                       import Mathlib.Tactic.NormNum

SLM(x) ->                                            theorem integral_eval :
                                                       (1792 / 10000 : ℝ) = 0.1792 := by
{                                                      norm_num
  [DRAFT]  Use Gamma identity
           Γ(7)/(Γ(4)Γ(3))                           Verification:
           Simplify polynomial                       symbolically_verified
           Integrate over [0,0.4]
                                                     Final Answer:
  [SKETCH] H(x) = {Gamma values known,               0.1792
                   integral valid}

  [PROVE]  phi(x):
           ∫0^{0.4} ... dy = 0.1792
}
```

Figure 8: Prompt examples of calculus illustrating stage-specific instructions for mapping an input problem x to $(r, \mathcal{H}(x), \phi(x))$ via a 1B-3B-parameter SLM, consistent with Equation (1).

```
Instruction:  Given a scientific problem x, generate structured reasoning and a formal specification.

Example (Differential Eq.):                          Formal Verification (Lean):

x: 4y'' + 12y' + 9y = 0,                             import Mathlib.Data.Real.Basic
y(0)=1, y'(0)=-4.                                    import Mathlib.Tactic.NormNum
Find where y(t)=0.

                                                     theorem ode_root :
SLM(x) ->                                              (1 - (5/2 : ℝ) * (2/5)) *
                                                         Real.exp (-(3/2 : ℝ) * (2/5)) = 0 := by
{                                                      norm_num
  [DRAFT]  Char eq: 4r^2+12r+9=0
           (2r+3)^2=0 -> r=-3/2                      Verification:
           y=(C1+C2 t)e^{-3t/2}                      symbolically_verified
           Apply ICs
                                                     Final Answer:
  [SKETCH] H(x) = {repeated root,                    0.4
                   ICs given}

  [PROVE]  phi(x):
           y(t)=(1-5t/2)e^{-3t/2}
           \land y(2/5)=0
}
```

Figure 9: Prompt examples of differential equations illustrating stage-specific instructions for mapping an input problem x to $(r, \mathcal{H}(x), \phi(x))$ via a 1B-3B-parameter SLM, consistent with Equation (1).

```
Instruction:  Given a scientific problem x, generate structured reasoning and a formal specification.

Example (Statistics):                                Formal Verification (Lean):

x: μ = 54.030, σ = 5.8,                              import Mathlib.Data.Real.Basic
n = 47. Find                                         import Mathlib.Tactic.NormNum
P(52.761 <= Xbar <= 54.453).
                                                     theorem prob_sample_mean :
SLM(x) ->                                              (52.761 : ℝ) <= 54.453 := by
                                                       norm_num
{
  [DRAFT]  z = (x - μ)/(σ/sqrt(n))                   Verification:
           Compute z1, z2                            symbolically_verified
           Use normal table
                                                     Final Answer:
  [SKETCH] H(x) = {μ = 54.030,                       0.6247
                   σ = 5.8,
                   n = 47,
                   bounds given}

  [PROVE]  phi(x):
           P(52.761 <= Xbar <= 54.453) = 0.6247
}
```

Figure 10: Prompt examples of statistics illustrating stage-specific instructions for mapping an input problem x to $(r, \mathcal{H}(x), \phi(x))$ via a 1B-3B-parameter SLM, consistent with Equation (1).

```
Instruction:  Given a scientific problem x, generate structured reasoning and a formal specification.

Example (Thermodynamics):                          Formal Verification (Lean):

x: Mercury column (10 m),                          import Mathlib.Data.Real.Basic
rho = 13.6 g/cm^3.                                 import Mathlib.Tactic.NormNum
Delta H_f = 2.292 kJ/mol,
T = 234.3 K.                                       theorem mercury_temp :
Find freezing temperature.                           (234.3 : ℝ) = 234.3 := by
                                                     norm_num
SLM(x) ->
                                                   Verification:
{                                                  symbolically_verified
  [DRAFT]  P = rho*g*h
           Use Clapeyron eqn                       Final Answer:
           Relate P and T                          234.3 K

  [SKETCH] H(x) = {rho = 13.6,
                   h = 10 m,
                   Delta H = 2.292,
                   T0 = 234.3}

  [PROVE]  phi(x):
           T = T0
           \land T = 234.3
}
```

Figure 11: Prompt examples of thermodynamics illustrating stage-specific instructions for mapping an input problem x to $(r, \mathcal{H}(x), \phi(x))$ via a 1B-3B-parameter SLM, consistent with Equation (1).

```
Instruction:  Given a scientific problem x, generate structured reasoning and a formal specification.

Example (Quantum Mechanics):                       Formal Verification (Lean):

x: Electron localized within                       import Mathlib.Data.Real.Basic
Delta x = 20 pm.                                   import Mathlib.Tactic.NormNum
Find uncertainty in speed.
                                                   theorem uncertainty_speed :
SLM(x) ->                                            (1.054e-34 : ℝ) /
                                                       (2 * 9.109e-31 * 20e-12) > 0 := by
{                                                    norm_num
  [DRAFT]  Delta x Delta p >= hbar/2
           Delta p = m Delta v                     Verification:
           Solve for Delta v                       symbolically_verified

  [SKETCH] H(x) = {Delta x = 20 pm,                Final Answer:
                   m = 9.109e-31,                  2.90e6 m/s
                   hbar = 1.054e-34}

  [PROVE]  phi(x):
           Delta v =
           hbar/(2*m*Delta x)
           \land Delta v > 0
}
```

Figure 12: Prompt examples of quantum mechanics illustrating stage-specific instructions for mapping an input problem x to $(r, \mathcal{H}(x), \phi(x))$ via a 1B-3B-parameter SLM, consistent with Equation (1).

```
Instruction:  Given a scientific problem x, generate structured reasoning and a formal specification.

Example (Geometry):                                Formal Verification (Lean):

x: Ball of string radius                           import Mathlib.Data.Real.Basic
r approx 2 m. Estimate total                       import Mathlib.Tactic.Ring
length L (order of magnitude).
                                                   theorem string_length :
SLM(x) ->                                            (4/3 : ℝ) * Real.pi * r^3 =
                                                       Real.pi * r_s^2 * L := by
{                                                    ring
  [DRAFT]  V = (4/3)pi*r^3
           Relate volume to L                      Verification:
           Estimate magnitude                      symbolically_verified

  [SKETCH] H(x) = {r approx 2 m,                   Final Answer:
                   spherical volume,               L = 4*r^3/(3*r_s^2)
                   string cross-section r_s}

  [PROVE]  phi(x):
           (4/3)pi*r^3 = pi*r_s^2*L
}
```

Figure 13: Prompt examples of geometry illustrating stage-specific instructions for mapping an input problem x to $(r, \mathcal{H}(x), \phi(x))$ via a 1B-3B-parameter SLM, consistent with Equation (1).

```
Instruction:  Given a scientific problem x, generate structured reasoning and a formal specification.

Example (Linear Algebra):                          Formal Verification (Lean):

x: y = (-1+i, 2, 3-i).                             import Mathlib.Data.Real.Basic
Find (y,y).                                        import Mathlib.Tactic.NormNum

SLM(x) ->                                          theorem dot_product :
                                                     ((-1)^2 + 1^2) + 4 +
{                                                    (3^2 + (-1)^2) = (16 : ℝ) := by
  [DRAFT]   (y,y) = Sum |y_i|^2                      norm_num
            = |-1+i|^2 + 2^2 + |3-i|^2
                                                   Verification:
  [SKETCH] H(x) = {y1 = -1+i,                      symbolically_verified
                   y2 = 2,
                   y3 = 3-i}                       Final Answer:
                                                   16
  [PROVE]  phi(x):
           (-1)^2 + 1^2 + 2^2 +
           3^2 + (-1)^2 = 16
}
```

Figure 14: Prompt examples of linear algebra illustrating stage-specific instructions for mapping an input problem x to $(r, \mathcal{H}(x), \phi(x))$ via a 1B-3B-parameter SLM, consistent with Equation (1).

Table 7: Domain-wise performance on LEANDATA using Phi-2. We compare +Trained (structured supervision) and +RL+Verification (full method) under zero-shot (ZS) and few-shot (FS) prompting.

	Zero-shot (ZS)						Few-shot (FS)						Compute	
Domain	VA	FR	SC	AC	E2E	Avg	VA	FR	SC	AC	E2E	Avg	Tok/s	Mem
+Trained (Ours w/o Verification RL)														
Physics	25.0	21.4	67.1	52.8	35.0	40.3	31.9	26.7	72.4	58.8	43.5	46.7	140	22.1
Chemistry	23.8	20.3	65.9	51.6	33.9	39.1	30.3	24.8	71.3	57.4	40.4	44.8	145	23.1
Probability	24.5	20.9	66.5	52.2	34.4	39.7	31.0	25.8	71.9	58.0	41.1	45.6	**146**	22.3
Calculus	27.7	23.6	**69.6**	55.1	38.4	42.9	34.6	28.7	**74.8**	60.8	46.7	49.1	138	**21.2**
Differential Equations	26.1	22.3	68.3	53.8	36.2	41.3	32.7	27.3	73.4	59.6	42.8	47.2	143	21.9
Statistics	24.2	20.7	66.4	51.9	34.1	39.5	30.7	25.4	71.7	57.6	40.8	45.2	139	21.7
Thermodynamics	23.7	20.1	65.7	51.4	33.5	38.9	30.1	24.7	71.1	57.1	40.1	44.6	144	23.6
Quantum Mechanics	23.1	19.7	65.1	50.9	33.0	38.4	29.5	24.2	70.6	56.6	39.4	44.1	142	23.4
Geometry	33.3	33.3	66.7	**66.7**	33.3	46.7	33.3	33.3	**66.7**	**66.7**	33.3	46.7	137	22.6
Linear Algebra	**50.0**	**50.0**	50.0	50.0	**50.0**	**50.0**	**50.0**	**50.0**	50.0	50.0	**50.0**	**50.0**	141	22.8
+RL + Verification (Full Method)														
Physics	39.6	33.1	77.0	69.3	49.8	53.8	45.2	38.6	80.4	74.1	56.1	58.9	139.9	28.4
Chemistry	38.2	32.0	75.7	68.1	48.5	52.5	43.0	36.6	79.1	72.4	53.7	57.0	135.6	27.8
Probability	39.1	32.8	76.4	69.0	49.2	53.3	44.2	37.5	79.9	73.6	54.9	58.0	**140.1**	29.2
Calculus	43.7	37.2	80.3	**73.2**	54.9	57.9	48.0	40.7	82.2	**76.3**	59.0	61.2	136.8	28.7
Differential Equations	40.9	34.5	78.2	71.1	51.2	55.2	46.0	39.2	81.2	75.0	56.9	59.7	138.3	27.9
Statistics	38.8	32.4	76.1	68.5	48.9	52.9	43.5	37.1	79.6	73.0	54.4	57.5	136.2	**27.5**
Thermodynamics	38.1	31.7	75.5	67.8	48.2	52.3	42.6	36.3	78.8	72.1	53.3	56.6	139.4	28.9
Quantum Mechanics	37.3	31.1	74.9	67.3	47.5	51.6	41.8	35.6	78.2	71.6	52.6	56.0	137.8	28.3
Geometry	**66.7**	**66.7**	66.7	66.7	**66.7**	**66.7**	66.7	**66.7**	**100.0**	66.7	66.7	73.3	137.1	27.6
Linear Algebra	50.0	50.0	**100.0**	50.0	50.0	60.0	**100.0**	50.0	100.0	50.0	**100.0**	**80.0**	138.7	28.1

Domain-Wise Analysis. Tables 7-10 present domain-wise results for Phi-2, TinyLlama, Qwen2-1.5B, and DeepSeek-1.3B, highlighting the heterogeneous difficulty of LEANDATA across scientific domains. Under +Supervised training, E2E across the eight larger domains ranges from 26.1%-39.4% and 30.8%-47.2% for zero-shot and few-shot prompting, respectively. In contrast, +RL+Verification generally improves E2E across domains and prompting settings, with corresponding ranges of 38.9%-54.9% and 45.8%-60.2%. These improvements are generally accompanied by gains in VA and FR. For example, in physics, Phi-2 improves from $35.0\% \rightarrow 49.8\%$ and $43.5\% \rightarrow 56.1\%$ under zero-shot and few-shot prompting, respectively, while Qwen2-1.5B improves from $37.5\% \rightarrow 52.2\%$ and $44.8\% \rightarrow 57.1\%$. For calculus, DeepSeek-Coder-1.3B improves from $39.4\% \rightarrow 54.8\%$ (zero-shot) and $47.2\% \rightarrow 60.2\%$ (few-shot). SC and AC also generally increase under +RL+Verification, indicating improved consistency among r, $\mathcal{H}(x)$, and $\phi(x)$. Geometry and Linear Algebra contain only 3 and 2 instances, respectively, making their percentages highly sensitive to individual examples and less representative. These gains are achieved with comparable inference throughput ($\sim$ 135-146 Tok/s) and moderate memory requirements ($\sim$ 21-29 GB) across models and domains.

Table 8: Domain-wise performance on LEANDATA using TinyLlama. We compare +Trained (structured supervision) and +RL+Verification (full method) under zero-shot (ZS) and few-shot (FS) prompting.

	Zero-shot (ZS)						**Few-shot (FS)**						**Compute**	
Domain	VA	FR	SC	AC	E2E	Avg	VA	FR	SC	AC	E2E	Avg	Tok/s	Mem
+Trained (Ours w/o Verification RL)														
Physics	18.9	15.6	59.7	45.5	27.4	33.4	24.1	19.8	66.0	50.4	32.6	38.8	**146**	21.9
Chemistry	18.1	14.9	58.8	44.8	26.7	32.7	23.5	19.2	65.2	49.7	31.5	38.2	139	23.4
Probability	19.2	15.8	59.4	45.2	27.2	33.4	24.2	19.7	65.7	50.1	32.2	38.7	137	21.7
Calculus	20.4	16.8	**61.0**	46.5	28.6	34.7	25.2	20.6	**67.0**	51.2	33.8	39.6	141	22.1
Differential Equations	19.8	16.3	60.5	46.1	28.1	34.2	24.7	20.2	66.5	50.8	33.1	39.2	138	**21.2**
Statistics	18.5	15.2	59.1	45.0	27.0	32.9	23.9	19.5	65.6	49.9	31.9	38.5	144	23.1
Thermodynamics	18.0	14.8	58.5	44.6	26.5	32.5	23.4	19.1	64.9	49.5	31.2	37.9	140	22.3
Quantum Mechanics	17.6	14.5	58.0	44.2	26.1	32.1	22.9	18.7	64.4	49.0	30.8	37.2	143	23.6
Geometry	33.3	33.3	33.3	33.3	33.3	33.3	33.3	33.3	66.7	33.3	33.3	40.0	145	22.6
Linear Algebra	**50.0**	**50.0**	50.0	**50.0**	**50.0**	**50.0**	**50.0**	**50.0**	50.0	**50.0**	**50.0**	**50.0**	142	22.8
+RL + Verification (Full Method)														
Physics	31.4	26.5	70.0	62.6	40.4	46.2	36.9	31.1	74.4	67.7	47.5	51.3	137.1	28.4
Chemistry	30.7	25.9	69.2	61.9	39.7	45.5	36.1	30.4	73.5	66.9	46.7	50.7	136.8	**27.5**
Probability	31.9	26.9	70.5	63.1	41.0	46.7	37.2	31.3	74.9	68.2	48.1	51.6	139.4	28.3
Calculus	33.5	28.3	71.4	64.0	42.4	47.9	39.1	32.9	75.5	69.0	49.7	53.2	135.6	28.1
Differential Equations	32.9	27.8	70.9	63.5	41.8	47.4	38.5	32.4	75.0	68.6	49.1	52.7	138.3	29.2
Statistics	31.1	26.2	69.7	62.2	40.1	45.9	36.6	30.8	74.1	67.5	47.2	51.2	**140.1**	27.8
Thermodynamics	30.5	25.7	69.0	61.6	39.5	45.3	35.9	30.2	73.4	66.8	46.5	50.6	137.8	27.6
Quantum Mechanics	29.9	25.2	68.4	61.0	38.9	44.7	35.3	29.7	72.8	66.2	45.8	49.9	136.2	28.9
Geometry	**66.7**	**66.7**	**66.7**	**66.7**	**66.7**	**66.7**	66.7	**66.7**	**100.0**	**66.7**	66.7	73.3	139.9	28.7
Linear Algebra	50.0	50.0	100.0	50.0	50.0	60.0	**100.0**	50.0	100.0	50.0	**100.0**	**80.0**	138.7	27.9

Table 9: Domain-wise performance on LEANDATA using Qwen2-1.5B. We compare +Trained (structured supervision) and +RL+Verification (full method) under zero-shot (ZS) and few-shot (FS) prompting.

	Zero-shot (ZS)						Few-shot (FS)						Compute	
Domain	VA	FR	SC	AC	E2E	Avg	VA	FR	SC	AC	E2E	Avg	Tok/s	Mem
+Trained (Ours w/o Verification RL)														
Physics	26.5	22.6	68.3	53.8	37.5	41.7	33.5	27.8	74.0	60.0	44.8	48.0	141	22.6
Chemistry	25.4	21.6	67.4	52.9	36.4	40.7	32.1	26.6	73.1	59.1	43.5	46.9	**146**	21.9
Probability	26.1	22.2	68.0	53.4	37.1	41.4	33.0	27.4	73.7	59.7	44.2	47.6	137	23.4
Calculus	28.5	24.3	**70.2**	55.1	39.0	43.4	35.5	29.5	**75.5**	61.5	46.2	49.7	144	22.1
Differential Equations	27.7	23.6	69.6	54.6	38.4	42.8	34.7	28.8	74.9	60.8	45.7	49.0	139	**21.2**
Statistics	25.8	21.9	67.8	53.2	36.9	41.1	32.6	27.1	73.5	59.5	43.9	47.3	142	23.6
Thermodynamics	25.2	21.4	67.1	52.7	36.2	40.5	32.0	26.5	72.9	59.0	43.3	46.7	138	22.3
Quantum Mechanics	24.7	21.0	66.6	52.2	35.8	40.1	31.4	26.0	72.4	58.5	42.8	46.2	140	21.7
Geometry	33.3	33.3	66.7	**66.7**	33.3	46.7	33.3	33.3	66.7	**66.7**	33.3	46.7	145	22.8
Linear Algebra	**50.0**	**50.0**	50.0	50.0	**50.0**	**50.0**	**50.0**	**50.0**	50.0	50.0	**50.0**	**50.0**	143	23.1
+RL + Verification (Full Method)														
Physics	41.2	35.0	78.2	71.0	52.2	55.5	46.3	39.3	81.0	74.6	57.1	59.7	138.7	28.7
Chemistry	40.2	34.1	77.4	70.2	51.3	54.6	45.3	38.5	80.4	74.0	56.1	58.9	137.1	**27.5**
Probability	41.0	34.8	78.0	70.7	52.0	55.3	46.1	39.1	80.8	74.4	56.8	59.5	136.8	28.3
Calculus	43.5	36.9	80.0	**72.5**	54.1	57.4	48.0	40.8	82.6	76.2	59.1	61.3	139.9	29.2
Differential Equations	42.8	36.3	79.5	72.0	53.4	56.8	47.2	40.1	82.1	75.7	58.3	60.6	135.6	27.9
Statistics	40.7	34.5	77.7	70.4	51.7	55.0	45.8	38.9	80.7	74.0	56.4	59.2	**140.1**	28.1
Thermodynamics	40.0	33.9	77.0	69.8	51.0	54.3	45.0	38.2	80.1	73.4	55.7	58.4	138.3	28.9
Quantum Mechanics	39.4	33.4	76.5	69.3	50.4	53.8	44.4	37.7	79.5	72.8	54.9	57.9	136.2	27.6
Geometry	**66.7**	**66.7**	**100.0**	66.7	**66.7**	**73.3**	66.7	**66.7**	**100.0**	**66.7**	**66.7**	73.3	137.8	28.4
Linear Algebra	50.0	50.0	100.0	50.0	50.0	60.0	**100.0**	50.0	100.0	50.0	100.0	**80.0**	139.4	27.8

Table 10: Domain-wise performance on LEANDATA using DeepSeek-Coder-1.3B. We compare +Trained (structured supervision) and +RL+Verification (full method) under zero-shot (ZS) and few-shot (FS) prompting.

	Zero-shot (ZS)						Few-shot (FS)						Compute	
Domain	VA	FR	SC	AC	E2E	Avg	VA	FR	SC	AC	E2E	Avg	Tok/s	Mem
+Trained (Ours w/o Verification RL)														
Physics	27.0	23.0	69.5	54.8	38.2	42.5	34.0	28.2	75.0	61.0	45.7	48.8	143	23.6
Chemistry	25.9	22.0	68.6	53.9	37.1	41.5	32.7	27.2	74.1	60.2	44.3	47.7	137	**21.2**
Probability	26.6	22.6	69.2	54.4	37.8	42.1	33.4	27.8	74.7	60.7	45.1	48.3	140	22.8
Calculus	29.0	24.7	**71.4**	56.1	39.4	44.1	36.0	30.0	**76.4**	62.4	47.2	50.4	145	22.1
Differential Equations	28.2	24.0	70.8	55.6	38.8	43.5	35.2	29.3	75.8	61.8	46.6	49.7	139	23.4
Statistics	26.3	22.3	69.0	54.2	37.5	41.9	33.1	27.6	74.5	60.6	44.8	48.1	**146**	21.7
Thermodynamics	25.7	21.8	68.3	53.7	36.8	41.3	32.5	27.1	73.9	60.1	44.2	47.6	142	22.3
Quantum Mechanics	25.2	21.4	67.8	53.2	36.4	40.8	31.9	26.6	73.3	59.6	43.7	47.0	138	21.9
Geometry	33.3	33.3	66.7	**66.7**	33.3	46.7	33.3	33.3	66.7	**66.7**	33.3	46.7	141	23.1
Linear Algebra	**50.0**	**50.0**	50.0	50.0	**50.0**	**50.0**	**50.0**	**50.0**	50.0	50.0	**50.0**	**50.0**	144	22.6
+RL + Verification (Full Method)														
Physics	42.0	35.4	79.2	71.8	52.8	56.2	48.0	41.2	82.2	75.7	58.6	61.1	136.8	28.1
Chemistry	40.9	34.6	78.4	71.0	51.8	55.3	46.2	39.8	81.5	75.0	57.5	60.0	139.4	**27.5**
Probability	41.7	35.1	79.0	71.5	52.5	56.0	47.0	40.7	81.9	75.5	58.2	60.7	137.1	28.7
Calculus	44.2	37.4	81.0	**73.4**	54.8	58.2	49.7	42.8	83.4	**77.1**	60.2	62.6	138.3	27.6
Differential Equations	43.5	36.9	80.5	72.9	54.1	57.6	48.9	42.1	82.9	76.6	59.5	62.0	135.6	28.9
Statistics	41.4	34.9	78.7	71.3	52.2	55.7	46.7	40.4	81.7	75.1	57.9	60.4	**140.1**	27.9
Thermodynamics	40.7	34.3	78.0	70.6	51.5	55.0	45.9	39.7	81.1	74.4	57.2	59.7	139.9	28.3
Quantum Mechanics	40.1	33.8	77.5	70.1	50.9	54.5	45.2	39.1	80.6	73.8	56.5	59.0	136.2	29.2
Geometry	**66.7**	**66.7**	**100.0**	66.7	**66.7**	**73.3**	66.7	**66.7**	**100.0**	66.7	66.7	73.3	137.8	27.8
Linear Algebra	50.0	50.0	100.0	50.0	50.0	60.0	**100.0**	50.0	100.0	50.0	**100.0**	**80.0**	138.7	28.4

Ablation Analysis. Tables 11-14 present ablation results across four SLMs under zero-shot and few-shot prompting, showing consistent trends across model backbones. Starting from Base, few-shot E2E ranges from 26.8%-40.2%, while structured supervision improves E2E (e.g., Phi-2: 36.9% $\rightarrow$ 42.3%, Qwen2-1.5B: 39.5% $\rightarrow$ 45.9%). Adding RL or formal verification individually yields further gains, with formal verification generally providing stronger improvements in formal validity and E2E success (e.g., DeepSeek-1.3B: 46.7% $\rightarrow$ 51.2% under +Ver.). The full method (+RL+Ver.) achieves the highest E2E across the four backbones, reaching 49.7%-54.0% under zero-shot and 47.3%-58.6% under few-shot prompting, with FS E2E reaching 55.6%, 47.3%, 57.8%, and 58.6% for Phi-2, TinyLlama, Qwen2-1.5B, and DeepSeek-1.3B, respectively. Removing DSP, structural checks, or alignment checks consistently reduces SC/AC and E2E, highlighting the contribution of structured decomposition and cross-stage consistency. Removing counterexample generation sharply reduces FR (e.g., Phi-2: 38.1% $\rightarrow$ 24.3%, Qwen2-1.5B: 39.7% $\rightarrow$ 26.7%, DeepSeek-1.3B: 40.3% $\rightarrow$ 26.9%), confirming its role in counterexample-based falsification. Re-

moving closed-loop execution further reduces VA and E2E despite occasional increases in FR, indicating that execution-aware feedback is important for formal verification and overall end-to-end success.

Table 11: Ablation analysis on LEANDATA using Phi-2. We evaluate the contribution of key components under zero-shot (ZS) and few-shot (FS) prompting. Abbreviations: DSP = Draft–Sketch–Prove Extensions; RL = Reinforcement Learning; Ver. = Formal Verification; Struct. = Structural Checks; Align. = Alignment Checks; CEx. = Counterexample Generation; CL = Closed-Loop Execution.

	Zero-shot (ZS)						Few-shot (FS)						Compute	
Setting	VA	FR	SC	AC	E2E	Avg	VA	FR	SC	AC	E2E	Avg	Tok/s	Mem
Base	21.4	18.2	62.5	48.1	29.6	36.0	27.8	22.6	68.3	53.7	36.9	41.9	**151.3**	**21.4**
+ Super. (w/ DSP, w/o RL, w/o Ver.)	24.9	21.3	66.8	52.6	34.8	40.1	31.5	26.2	72.1	58.4	42.3	46.1	147.8	22.9
+ RL (w/ DSP, w/ RL, w/o Ver.)	28.1	24.2	69.5	56.7	37.9	43.3	34.8	28.6	74.6	61.9	45.6	49.1	143.6	24.1
+ Ver. (w/ DSP, w/o RL, w/ Ver.)	29.6	25.8	70.8	58.9	39.4	45.0	36.2	30.9	75.9	64.3	47.8	51.0	145.2	23.5
+ RL+Ver. (w/ DSP, w/ RL, w/ Ver.)	**39.6**	**33.2**	**76.8**	**69.1**	**49.7**	**53.7**	**44.8**	**38.1**	**80.2**	**73.5**	**55.6**	**58.4**	140.4	27.8
w/o DSP (w/ RL, w/ Ver.)	28.4	24.5	66.9	56.2	38.7	43.0	34.9	28.3	71.6	60.5	44.8	48.0	142.7	25.6
w/o Struct. (w/ DSP, w/ RL, w/ Ver.)	29.7	26.1	64.8	57.1	39.8	43.5	36.2	30.0	68.5	61.3	45.6	48.3	141.9	26.2
w/o Align. (w/ DSP, w/ RL, w/ Ver.)	30.6	27.4	70.5	58.0	40.8	45.5	37.4	31.3	74.6	62.1	47.3	50.5	**144.1**	24.8
w/o CEx. (w/ DSP, w/ RL, w/ Ver.)	**34.9**	20.5	**75.9**	**68.3**	**44.8**	**48.9**	**40.6**	24.3	**79.5**	**71.6**	**49.6**	53.1	143.2	25.1
w/o CL (w/ DSP, w/ RL, w/ Ver.)	32.6	**28.3**	73.7	66.1	42.9	48.7	38.9	**33.2**	77.9	69.4	48.3	**53.5**	142.3	**21.1**

Table 12: Ablation analysis on LEANDATA using TinyLlama. We evaluate the contribution of key components under zero-shot (ZS) and few-shot (FS) prompting. Abbreviations: DSP = Draft–Sketch–Prove Extensions; RL = Reinforcement Learning; Ver. = Formal Verification; Struct. = Structural Checks; Align. = Alignment Checks; CEx. = Counterexample Generation; CL = Closed-Loop Execution.

	Zero-shot (ZS)						Few-shot (FS)						Compute	
Setting	VA	FR	SC	AC	E2E	Avg	VA	FR	SC	AC	E2E	Avg	Tok/s	Mem
Base	15.2	12.4	55.1	41.3	21.7	29.1	19.6	15.8	60.7	45.9	26.8	33.8	145.2	**21.4**
+ Super. (w/ DSP, w/o RL, w/o Ver.)	18.7	15.4	59.8	45.6	27.5	33.4	23.8	19.6	65.9	50.2	32.1	38.3	**151.3**	24.1
+ RL (w/ DSP, w/ RL, w/o Ver.)	21.4	17.8	62.6	49.1	30.4	36.3	27.1	22.2	68.3	53.9	36.8	41.7	140.4	23.5
+ Ver. (w/ DSP, w/o RL, w/ Ver.)	22.6	19.2	63.9	51.2	31.8	37.7	28.5	24.1	69.7	56.1	38.7	43.4	143.6	22.9
+ RL+Ver. (w/ DSP, w/ RL, w/ Ver.)	**31.2**	**26.4**	**70.1**	**62.7**	**40.2**	**46.1**	**36.7**	**30.9**	**74.5**	**67.8**	**47.3**	**51.4**	147.8	27.8
w/o DSP (w/ RL, w/ Ver.)	22.8	19.5	61.2	51.3	29.6	36.9	27.9	22.6	66.0	55.4	34.1	41.2	143.2	26.2
w/o Struct. (w/ DSP, w/ RL, w/ Ver.)	23.7	20.7	59.1	52.0	30.4	37.2	28.8	23.7	63.5	56.0	34.9	41.4	142.3	**21.1**
w/o Align. (w/ DSP, w/ RL, w/ Ver.)	24.5	21.6	63.8	52.7	31.3	38.8	29.7	24.7	68.0	56.6	36.2	43.0	**144.1**	25.1
w/o CEx. (w/ DSP, w/ RL, w/ Ver.)	**27.8**	15.9	**68.7**	**61.8**	**33.9**	**41.6**	**32.4**	18.6	**72.3**	**65.2**	**38.5**	45.4	141.9	24.8
w/o CL (w/ DSP, w/ RL, w/ Ver.)	26.1	**22.5**	66.9	59.6	32.5	41.5	31.0	**26.9**	70.8	63.0	37.4	**45.8**	142.7	25.6

Table 13: Ablation analysis on LEANDATA using Qwen2-1.5B. We evaluate the contribution of key components under zero-shot (ZS) and few-shot (FS) prompting. Abbreviations: DSP = Draft–Sketch–Prove Extensions; RL = Reinforcement Learning; Ver. = Formal Verification; Struct. = Structural Checks; Align. = Alignment Checks; CEx. = Counterexample Generation; CL = Closed-Loop Execution.

	Zero-shot (ZS)						Few-shot (FS)						Compute	
Setting	VA	FR	SC	AC	E2E	Avg	VA	FR	SC	AC	E2E	Avg	Tok/s	Mem
Base	23.7	20.1	64.2	50.5	32.8	38.3	29.9	24.3	70.4	56.8	39.5	44.2	143.6	**21.4**
+ Super. (w/ DSP, w/o RL, w/o Ver.)	27.9	23.8	69.5	55.1	38.6	43.0	34.6	28.5	74.8	60.7	45.9	48.9	147.8	23.5
+ RL (w/ DSP, w/ RL, w/o Ver.)	30.8	26.3	72.2	59.0	41.5	46.0	37.9	31.0	77.1	64.5	48.6	51.8	140.4	24.1
+ Ver. (w/ DSP, w/o RL, w/ Ver.)	32.4	27.9	73.5	61.3	43.0	47.6	39.4	33.5	78.6	66.8	50.4	53.7	**151.3**	22.9
+ RL+Ver. (w/ DSP, w/ RL, w/ Ver.)	**41.8**	**35.6**	**78.9**	**71.8**	**52.9**	**56.2**	**46.9**	**39.7**	**81.6**	**75.1**	**57.8**	**60.2**	145.2	27.8
w/o DSP (w/ RL, w/ Ver.)	30.5	26.3	69.5	58.7	41.5	45.3	36.8	30.0	74.3	63.5	48.6	50.6	141.9	25.1
w/o Struct. (w/ DSP, w/ RL, w/ Ver.)	31.8	28.0	67.2	59.6	42.6	45.8	38.1	31.9	71.5	64.4	49.7	51.1	143.2	**21.1**
w/o Align. (w/ DSP, w/ RL, w/ Ver.)	32.7	29.2	73.2	60.3	43.7	47.8	39.3	33.1	77.6	65.1	51.2	53.3	142.7	24.8
w/o CEx. (w/ DSP, w/ RL, w/ Ver.)	**37.1**	22.4	**78.0**	**71.0**	**48.1**	**51.3**	**42.6**	26.7	**81.0**	**74.3**	**53.6**	55.6	142.3	26.2
w/o CL (w/ DSP, w/ RL, w/ Ver.)	34.8	**30.0**	75.9	68.2	45.9	50.9	41.2	**34.8**	80.1	72.0	51.7	**56.0**	**144.1**	25.6

Table 14: Ablation analysis on LEANDATA using DeepSeek-Coder-1.3B. We evaluate the contribution of key components under zero-shot (ZS) and few-shot (FS) prompting. Abbreviations: DSP = Draft–Sketch–Prove Extensions; RL = Reinforcement Learning; Ver. = Formal Verification; Struct. = Structural Checks; Align. = Alignment Checks; CEx. = Counterexample Generation; CL = Closed-Loop Execution.

	Zero-shot (ZS)						Few-shot (FS)						Compute	
Setting	VA	FR	SC	AC	E2E	Avg	VA	FR	SC	AC	E2E	Avg	Tok/s	Mem
Base	24.1	20.4	65.0	51.2	33.4	38.8	30.5	24.9	71.2	57.6	40.2	45.1	140.4	**21.4**
+ Super. (w/ DSP, w/o RL, w/o Ver.)	28.3	24.2	70.1	55.7	39.1	43.5	35.1	29.0	75.4	61.3	46.7	49.8	145.2	27.8
+ RL (w/ DSP, w/ RL, w/o Ver.)	31.3	26.8	72.8	59.6	42.0	46.5	38.4	31.5	77.8	65.1	49.3	52.4	143.6	23.5
+ Ver. (w/ DSP, w/o RL, w/ Ver.)	32.8	28.5	74.2	61.9	43.6	48.2	39.9	34.1	79.1	67.5	51.2	54.4	**151.3**	22.9
+ RL+Ver. (w/ DSP, w/ RL, w/ Ver.)	**42.5**	**36.2**	**79.5**	**72.4**	**54.0**	**56.9**	**47.6**	**40.3**	**82.2**	**75.8**	**58.6**	**60.8**	147.8	24.1
w/o DSP (w/ RL, w/ Ver.)	31.0	26.8	70.1	59.2	42.2	45.9	37.6	30.6	75.0	63.8	48.8	51.2	142.3	25.6
w/o Struct. (w/ DSP, w/ RL, w/ Ver.)	32.3	28.4	67.8	60.1	43.4	46.4	39.0	32.1	72.2	64.7	50.1	51.7	141.9	**21.1**
w/o Align. (w/ DSP, w/ RL, w/ Ver.)	33.2	29.6	73.7	60.9	44.5	48.4	40.2	33.8	78.1	65.6	51.7	53.9	143.2	24.8
w/o CEx. (w/ DSP, w/ RL, w/ Ver.)	**37.6**	22.7	**78.6**	**71.6**	**48.9**	**51.9**	**43.3**	26.9	**81.6**	**74.9**	**54.4**	56.2	142.7	25.1
w/o CL (w/ DSP, w/ RL, w/ Ver.)	35.2	**30.4**	76.5	68.7	46.7	51.5	41.8	**35.2**	80.8	72.5	52.9	**56.6**	**144.1**	26.2

Table 15: Analysis of verification depth in LEANDATA under zero-shot (ZS) and few-shot (FS) evaluation. We compare numeric-only, equation-level, dimensional-consistent, symbolic-derivation, and premise-grounded verification across the same SLM backbones, where structured verification evaluates reasoning steps r, assumptions $\mathcal{H}(x)$, dimensional constraints, and formal specifications $\phi(x)$. Higher Avg is better.

Model	**Verification Strategy**	**ZS Avg**	**FS Avg**
	Phi-2		
Numeric-Only Verification	Final-value checks via `norm_num`	44.6	50.1
Equation-Level Verification	Verify reconstructed equations	50.3	55.2
Dimensional-Consistent Verification	Verify unit and dimensional consistency	51.4	56.1
Symbolic-Derivation Verification	Verify symbolic transformations	52.6	57.3
Premise-Grounded Verification	Verify $r \leftrightarrow \mathcal{H}(x) \leftrightarrow \phi(x)$	**53.7**	**58.4**
	TinyLlama		
Numeric-Only Verification	Final-value checks via `norm_num`	38.8	43.6
Equation-Level Verification	Verify reconstructed equations	43.4	48.2
Dimensional-Consistent Verification	Verify unit and dimensional consistency	44.1	49.0
Symbolic-Derivation Verification	Verify symbolic transformations	45.2	50.3
Premise-Grounded Verification	Verify $r \leftrightarrow \mathcal{H}(x) \leftrightarrow \phi(x)$	**46.1**	**51.4**
	Qwen2-1.5B		
Numeric-Only Verification	Final-value checks via `norm_num`	47.0	51.8
Equation-Level Verification	Verify reconstructed equations	52.6	57.4
Dimensional-Consistent Verification	Verify unit and dimensional consistency	53.4	58.2
Symbolic-Derivation Verification	Verify symbolic transformations	54.9	59.1
Premise-Grounded Verification	Verify $r \leftrightarrow \mathcal{H}(x) \leftrightarrow \phi(x)$	**56.2**	**60.2**
	Gemma-2B		
Numeric-Only Verification	Final-value checks via `norm_num`	48.2	52.9
Equation-Level Verification	Verify reconstructed equations	54.2	58.8
Dimensional-Consistent Verification	Verify unit and dimensional consistency	55.1	59.6
Symbolic-Derivation Verification	Verify symbolic transformations	56.5	60.5
Premise-Grounded Verification	Verify $r \leftrightarrow \mathcal{H}(x) \leftrightarrow \phi(x)$	**58.0**	**61.6**
	DeepSeek-1.3B		
Numeric-Only Verification	Final-value checks via `norm_num`	47.5	52.4
Equation-Level Verification	Verify reconstructed equations	53.5	58.1
Dimensional-Consistent Verification	Verify unit and dimensional consistency	54.3	58.9
Symbolic-Derivation Verification	Verify symbolic transformations	55.8	59.8
Premise-Grounded Verification	Verify $r \leftrightarrow \mathcal{H}(x) \leftrightarrow \phi(x)$	**57.2**	**60.8**

Verification Analysis. We compare numeric-only, equation-level, dimensional-consistent, symbolic-derivation, and premise-grounded formal verification across all five SLM backbones under zero-shot and few-shot evaluation (see Table 15). Numeric-only verification via `norm_num` yields weaker Avg performance (zero-shot: 38.8%-48.2%; few-shot: 43.6%-52.9%), while equation-level, dimensional-consistent, and symbolic-derivation verification provide consistent improvements. Premise-grounded formal verification, jointly evaluating reasoning steps r, assumptions $\mathcal{H}(x)$, and encoded formal specifications $\phi(x)$, achieves the strongest performance (zero-shot Avg: 46.1%-58.0%; few-shot Avg: 51.4%-61.6%), indicating that the reported gains are associated with structured verification of scientific derivations rather than trivial numeric equality checks.

Counterexample Analysis. We compare fixed numeric assertions, equation-constrained, physically-constrained, and symbolic scientific specifications across all five SLM backbones under

Table 16: Analysis of counterexample generation in LEANDATA under zero-shot (ZS) and few-shot (FS) evaluation. We compare fixed numeric, equation-constrained, physically-constrained, and symbolic scientific specifications, where structured specifications support falsification through equation inconsistency, assumption violation, dimensional consistency, state-variable constraints, and premise-grounded contradiction. Symbolic scientific specifications additionally evaluate physically valid counterexamples against executable hypotheses $\phi(x)$. Higher Avg is better.

Model	Specification Strategy	ZS Avg	FS Avg
Phi-2			
Fixed Numeric Assertions	Final-value specifications only	41.6	47.3
Equation-Constrained Specifications	Counterexamples over reconstructed equations	49.4	54.1
Physically-Constrained Specifications	Counterexamples with unit/state constraints	51.6	56.0
Symbolic Scientific Specifications	Counterexamples over r, $\mathcal{H}(x)$, $\phi(x)$	**54.2**	**58.4**
TinyLlama			
Fixed Numeric Assertions	Final-value specifications only	36.1	40.8
Equation-Constrained Specifications	Counterexamples over reconstructed equations	42.2	46.5
Physically-Constrained Specifications	Counterexamples with unit/state constraints	44.1	48.4
Symbolic Scientific Specifications	Counterexamples over r, $\mathcal{H}(x)$, $\phi(x)$	**46.2**	**51.4**
Qwen2-1.5B			
Fixed Numeric Assertions	Final-value specifications only	44.5	49.6
Equation-Constrained Specifications	Counterexamples over reconstructed equations	51.7	56.3
Physically-Constrained Specifications	Counterexamples with unit/state constraints	53.8	57.7
Symbolic Scientific Specifications	Counterexamples over r, $\mathcal{H}(x)$, $\phi(x)$	**56.2**	**60.2**
Gemma-2B			
Fixed Numeric Assertions	Final-value specifications only	46.1	50.8
Equation-Constrained Specifications	Counterexamples over reconstructed equations	53.3	57.6
Physically-Constrained Specifications	Counterexamples with unit/state constraints	55.6	59.2
Symbolic Scientific Specifications	Counterexamples over r, $\mathcal{H}(x)$, $\phi(x)$	**58.0**	**61.6**
DeepSeek-1.3B			
Fixed Numeric Assertions	Final-value specifications only	45.4	50.2
Equation-Constrained Specifications	Counterexamples over reconstructed equations	52.6	57.0
Physically-Constrained Specifications	Counterexamples with unit/state constraints	54.7	58.5
Symbolic Scientific Specifications	Counterexamples over r, $\mathcal{H}(x)$, $\phi(x)$	**57.2**	**60.8**

zero-shot and few-shot evaluation (see Table 16). Fixed numeric specifications yield weaker Avg performance (zero-shot: 36.1%-46.1%; few-shot: 40.8%-50.8%), while equation-constrained and physically-constrained specifications improve performance through reconstructed equations, dimensional consistency, unit constraints, and state-variable conditions. Symbolic scientific specifications achieve the strongest performance by jointly evaluating reasoning steps r, assumptions $\mathcal{H}(x)$, and executable hypotheses $\phi(x)$ (zero-shot Avg: 46.2%-58.0%; few-shot Avg: 51.4%-61.6%), with ground-truth false hypotheses constructed through controlled perturbations of equations, assumptions, units, and parameter constraints within executable specifications $\phi(x)$.

Domain Sensitivity Analysis. We compare low-resource domain evaluation, balanced resampling, merged-domain evaluation, and full-domain aggregation under zero-shot and few-shot settings (see Table 17). Tiny-domain evaluation yields higher variance (zero-shot: 7.6%-9.8%; few-shot: 6.9%-9.0%), while balanced resampling and merged-domain evaluation reduce variance, and full-domain aggregation yields the most stable estimates (zero-shot: 2.4%-3.6%; few-shot: 2.1%-3.1%), indicating consistent trends under balanced and aggregated evaluation settings.

Table 17: Domain-size sensitivity in LEANDATA under zero-shot (ZS) and few-shot (FS) evaluation. We compare low-resource domains, balanced resampling, merged-domain evaluation, and full-domain aggregation. Higher Avg and lower Var are better.

Model	Domain Evaluation Strategy	ZS Avg	ZS Var	FS Avg	FS Var
Phi-2 +RL+Verif					
Tiny Domains Only	Geometry/Linear Algebra subsets	47.8	8.9	52.1	8.1
Balanced Domain Resampling	Equalized domain sampling	51.2	4.6	55.9	4.1
Merged STEM Domains	Aggregate low-resource domains	52.4	3.8	57.1	3.3
Full-Domain Aggregation	All domains jointly evaluated	**53.7**	**2.9**	**58.4**	**2.5**
TinyLlama +RL+Verif					
Tiny Domains Only	Geometry/Linear Algebra subsets	40.1	9.8	44.5	9.0
Balanced Domain Resampling	Equalized domain sampling	43.2	5.3	47.8	4.7
Merged STEM Domains	Aggregate low-resource domains	44.4	4.5	49.1	4.0
Full-Domain Aggregation	All domains jointly evaluated	**46.1**	**3.6**	**51.4**	**3.1**
Qwen2-1.5B +RL+Verif					
Tiny Domains Only	Geometry/Linear Algebra subsets	49.6	8.1	54.3	7.5
Balanced Domain Resampling	Equalized domain sampling	53.1	4.2	57.4	3.9
Merged STEM Domains	Aggregate low-resource domains	54.5	3.5	58.9	3.1
Full-Domain Aggregation	All domains jointly evaluated	**56.2**	**2.7**	**60.2**	**2.4**
Gemma-2B +RL+Verif					
Tiny Domains Only	Geometry/Linear Algebra subsets	51.4	7.6	55.8	6.9
Balanced Domain Resampling	Equalized domain sampling	54.8	3.9	58.9	3.5
Merged STEM Domains	Aggregate low-resource domains	56.1	3.2	60.1	2.9
Full-Domain Aggregation	All domains jointly evaluated	**58.0**	**2.4**	**61.6**	**2.1**
DeepSeek-1.3B +RL+Verif					
Tiny Domains Only	Geometry/Linear Algebra subsets	50.8	7.9	55.1	7.2
Balanced Domain Resampling	Equalized domain sampling	54.1	4.0	58.3	3.6
Merged STEM Domains	Aggregate low-resource domains	55.4	3.4	59.5	3.0
Full-Domain Aggregation	All domains jointly evaluated	**57.2**	**2.5**	**60.8**	**2.2**

Keyword Definitions. In Table 5, *Criterion* denotes the specific dimension used to evaluate the encoded formal specification: *Equation Fidelity* measures whether the required equations are correctly encoded; *Constraint Fidelity* measures whether the problem constraints are preserved; *Variable & Unit Consistency* measures the correctness and consistency of variables and their units; *Semantic Fidelity* measures whether the encoded formal specification preserves the intended meaning, including boundary conditions; and *Lean Executability* measures whether the resulting specification is valid and executable in Lean. *Overall Faithfulness* summarizes these dimensions, while *IAA* denotes inter-annotator agreement.