Counterexample Generation via Per-Theorem Symbolic Verifiers: When Imitation Hurts and Reinforcement Repairs

Anonymous ACL submission

Abstract

Large language models often solve a theorem forward yet fail to disprove a closely related false one—a falsification gap that supervised fine-tuning does not close and can actively worsen. We frame counterexample generation as constrained witness emission against a deterministic per-theorem Python verifier, and construct a corpus of 4,707 false undergraduate-algebra and real-analysis conjectures, each paired with an independent verifier module and a witness schema. Training Qwen3-4B with SFT followed by GRPO under this oracle reveals an *imitation trap*: counterexample-only SFT collapses true-theorem recognition from 0.27 to 0.00 under the training prompt style, while RLVR with a sparse outcome-only reward not only repairs this but exceeds the base (to 0.65). Sparse and dense rewards yield statistically indistinguishable in-domain success (0.49 vs 0.50 Pass@1) yet diverge by 30 points on a held-out calibration probe, a dissociation we trace to the partial-credit term, which incentivises over-eager witness emission on false conjectures and transfers that bias to true ones. Our 4B RLVR model outperforms every evaluated 7B open-weights math specialist on the in-domain benchmark, remains competitive with six frontier commercial APIs, and transfers under unchanged chain-of-thought prompting to GSM8K (0.89), MATH-500 (0.65) and MMLU-college-math (0.69). We release the dataset, verifier modules, audit suite, and evaluation pipeline.

1 Introduction

Large Language Models (LLMs) excel at forward mathematical derivation, but struggle significantly with inverse reasoning and generating counterexamples to false claims. Recent benchmarks demonstrate that even frontier models fail at falsification tasks they can easily forward-solve (Sinha et al., 2025; Li et al., 2025). Furthermore, standard imitation learning (SFT) often exacerbates this by inducing format-biases rather than true logical reasoning (Shao et al., 2025).

The natural fix is to train counterexample generation directly, but two obstacles have so far prevented this from becoming a primary objective. First, large-scale counterexample data is scarce: the only widely cited counterexample benchmark, COUNTERMATH, contains 1,216 problems, and its evaluation is mediated by an LLM judge. Second, supervised fine-tuning (SFT) on counterexamples appears insufficient. Counterexample-only training data contains, by construction, only false statements, and we show below that imitation-only training transfers this distribution shift into the policy: a model trained purely to emit counterexamples loses the ability to recognise true statements, and confidently fabricates witnesses on theorems that are in fact true.

We attack both obstacles together. We build a deterministic, large-scale counterexample dataset by hypothesis-dropping over 6,748 undergraduate-level theorems and generating, for each `(theorem_id, dropped_hypothesis)` pair, an independent Python verifier module that returns one boolean per remaining hypothesis and one for the negated conclusion. The pipeline retains 4,707 records. Evaluation is fully symbolic: the policy emits a one line Python expression, and the verifier returns a boolean tuple.

We use this environment to train Qwen3-4B (Qwen Team, 2025) with SFT followed by GRPO under two reward shapes and to evaluate against five 7B open-weights math specialists, six commercial-API frontier models, and four external mathematics benchmarks. *In doing so, we uncover a pathology*: imitation-only training on counterexamples collapses the policy's ability to recognise true theorems, and only verifier-grounded RL with an outcome-only reward repairs it. Partial-credit dense rewards yield equal in-domain success but do not restore calibration; a dissociation that, to

our knowledge, has not been previously isolated.

Contributions. We make four main contributions. **(C1) The Imitation Trap Pathology.** We isolate a calibration failure mode in counterexample-only supervised fine-tuning: Pass@1 on true theorems collapses from 0.27 to 0.00 under the training prompt style, because the policy learns the output *form* of a counterexample without the negated-conclusion constraint. **(C2) Verifier-Grounded RLVR for Falsification.** We demonstrate that Reinforcement Learning with Verifiable Rewards (RLVR), guided by a deterministic symbolic execution interface, repairs this pathology. Furthermore, we reveal that sparse (outcome-only) rewards preserve logical calibration significantly better than dense (partial-credit) rewards. **(C3) Strong Empirical Performance & Transfer.** Our Qwen3-4B RLVR model achieves a 0.50 Pass@1 on in-domain falsification, outperforming all evaluated 7B open-weights math specialists and remaining competitive with frontier APIs, while demonstrating strong zero-shot transfer to standard forward-math tasks (MATH-500, GSM8K). **(C4) A Deterministic Falsification Benchmark and Data Preparation Pipeline.** To enable this training, we introduce and release a rigorous, programmatically verifiable dataset of 4,707 false algebraic conjectures, moving beyond the noisy LLM-as-a-judge paradigm.

The code, data, verifier modules, audit suite, and evaluation pipeline will be released upon acceptance.

2 Related Work

Counterexample and falsification benchmarks. The closest comparable resource for counterexample reasoning is COUNTERMATH (Li et al., 2025), a 1,216-problem benchmark of university-level counterexample-driven proofs evaluated with an LLM-as-judge protocol; the largest frontier model reported in that work achieves an F1 of 60.1. REFUTE (Sinha et al., 2025) measures counterexample success on algorithmic problems and reports a sharp asymmetry between forward solution rates (around 48 %) and counterexample success (below 9 %) for the same models. The most directly related training resource is *Learning to Disprove* (Li et al., 2026), which constructs roughly 575K hypothesis-dropping mutations of Lean 4 Mathlib theorems and uses Lean tactic checks as the verification signal. Our dataset is smaller than Li et al.'s but uses a Python-verifier substrate that is more faithful to the algebraic and real-analytic structures evaluated, runs in tens of milliseconds rather than seconds, and is paired with the witness-schema interface of §3.2. The Alchemy line of AST-mutation work (Wu et al., 2024) focuses on generating valid mutated *true* theorems for automated theorem proving and is complementary rather than competing.

Verifiable-reward reinforcement learning for mathematics. RLVR with GRPO has become the dominant post-training paradigm for forward mathematical reasoning, anchored by DeepSeek-R1 (Guo et al., 2025) and adopted in PRIME (Cui et al., 2025) and Eurus (Yuan et al., 2024). While recent studies debate whether RLVR installs new reasoning or merely elicits latent capabilities via format adaptation (Shao et al., 2025), our out-of-distribution transfer results (§6.5) suggest falsification training installs fundamentally transferable inverse-reasoning skills.

Process supervision and symbolic verifiers. Process reward models (PRMs), introduced for mathematical reasoning by Lightman et al. (2024) and extended by Math-Shepherd (Wang et al., 2023), and the FOVER pipeline (Kamoi et al., 2026), which mines step-level labels from Z3 and Isabelle traces, supply dense supervision at the step level for forward proving. Our per-theorem verifier sits between an outcome reward model and a PRM: it returns one boolean per remaining hypothesis and one for the negated conclusion, which gives the reward function (Eq. 1) a structured, audited signal without the human-annotation cost of a learned PRM. SymPy and Z3 (Meurer et al., 2017; De Moura and Bjørner, 2008) provide the underlying verification primitives.

The hallucination tax. A growing literature describes a calibration pathology induced by standard imitation-style training: TreeCut (Ouyang, 2025) demonstrates that LLMs continue to fabricate answers when problems are systematically modified to remove a critical constraint; SUM (Song et al., 2025) and related work report refusal-rate collapses of roughly 80 % under standard RFT on unanswerable-math benchmarks. Our imitation-trap finding (§6.2) is the cleanest measurement of this pathology in the counterexample-generation setting we are aware of, and the first to localise the mechanism, via the reason taxonomy of §6.3, to the partial-credit term of the reward function.

3 Dataset and Verification Environment

We frame counterexample generation as a constrained generation task: given a false conjecture of the form "if $H_1 \wedge \cdots \wedge H_k$ then Q", output a witness w such that all remaining hypotheses hold and Q fails on w. Our environment is built around three design choices: a per-theorem Python verifier compiled once and reused across all rollouts, a structured witness-schema interface that constrains the policy's output shape, and a conclusion-weighted dense reward whose sparse counterpart we use for ablation.

3.1 Source theorems and hypothesis dropping

We collected 6,748 unique undergraduate-level mathematical theorems spanning six algebraic subdomains (Group, Ring, Field, Number, Linear-Algebra and Real-Analysis). Each theorem is stored in a structured form with explicit *hypotheses* and *conclusion* fields. To construct false conjectures, we apply a hypothesis-dropping mutation over the source theorems. Because removing a hypothesis does not strictly guarantee the resulting statement is false or well-posed (e.g., if the original hypothesis was redundant or merely clarificatory), this initial mutation yields a noisy corpus of $\sim$13.5K candidates. To ensure rigorous quality, we apply an LLM-driven verification filter (Gemini 3.1 Pro): the teacher model must both classify the input mutated theorem as well-posed and non-trivial, and successfully generate a valid Python verifier, propose a counterexample, and the counterexample must evaluate to 'True' on the verifier without execution errors. We allow up to 5 retries with error-trace feedback. This strict programmatic filtering eliminates trivial, ill-posed, or still-true statements, distilling the corpus to 4,707 high-quality, verifier-backed records covering 3,485 unique theorems. We report per-subdomain counts in Table 1.

A sandboxed subprocess imports the verifier dynamically, builds the witness-evaluation namespace, evaluates the policy's witness expression string in that namespace, and calls `conditions(witness)`, which returns a list of booleans, one per remaining hypothesis followed by one for the negated conclusion. The policy and the oracle share *no code paths* beyond Python's `eval`; the verifier is fixed at the start of training and reused across all rollouts for that conjecture.

This is a deliberate departure from prior counterexample-evaluation pipelines: COUNTER-MATH (Li et al., 2025) relies on an LLM-as-judge protocol, and *Learning to Disprove* (Li et al., 2026) relies on a Lean tactic check. A Python module evaluated against a witness expression is intermediate: it is symbolic and decidable for the subdomains we target, fast (§3.5), and reproducible.

3.2 Witness-schema interface

We attach to each prompt a one-line *witness schema*: a short Python-shape description of an acceptable witness. Schemas are generated post-hoc by reading the verifier module and its canonical witness with a second teacher model (DeepSeek) and asking for a single-sentence description of the expected Python object shape, with one or two concrete examples. Sample schemas are shown in D.

Schemas are merged into the user prompt. We view this as a small but load-bearing methodological contribution: a structural prior on the output shape, generated automatically from the verifier and a canonical witness, is what allows for the practical RLVR effectiveness without altering either the reward function or the verifier.

3.3 Split construction

We construct three splits with per-domain stratified sampling and a fixed seed (42): the held-out test set (n=201), the SFT split (n=1,000), and the RLVR prompt set (n=3,506). Table 1 summarises the splits.

3.4 Reward function

The per-rollout reward computed by the oracle is

$$\begin{aligned} r = \alpha \cdot \frac{s_{\text{hyp}}}{n_{\text{hyp}}} + \beta \cdot q_{\neg} + \gamma \cdot \mathbb{1}[\text{format ok}] \\ -\delta \cdot \mathbb{1}[\text{gave up on false}] \\ -\varepsilon \cdot (1 - \mathbb{1}[\text{parse ok}]) \,, \end{aligned} \tag{1}$$

where $s_{\text{hyp}} = \sum_{i=1}^{n_{\text{hyp}}} \texttt{conditions}[i]$ counts satisfied hypotheses, $q_{\neg} = \texttt{conditions}[\texttt{neg_q_index}] \in \{0, 1\}$ is the negated-conclusion indicator, and the format / parse flags come from the structural and witness-evaluation checks. The penalty δ applies when the policy emits the literal sentinel `NONE` on a false conjecture under the Stage-1 training mode used throughout this paper.

Dense and sparse variants. We study two reward shapes. *Dense*: α=0.25, β=0.65, γ=0.10, δ=0.30, ε=1.00, range approximately $[-1, +1]$. *Sparse*:

$\alpha{=}\gamma{=}\delta{=}0, \beta{=}\varepsilon{=}1$, range $\{-1, 0, +1\}$. The same SFT checkpoint is used as the reference policy for both.

Conclusion-weighting bound. A structural property of Eq. (1) under dense weights is that any successful, formatting-compliant rollout ($\mathbb{1}$[format ok] $=$ $\mathbb{1}$[parse ok] $= 1$ and $\mathbb{1}$[gave up on false] $= 0$) that falls into the "hypothesis-only" failure mode ($s_{\text{hyp}} = n_{\text{hyp}}$, $q_{\neg} = 0$) is strictly dominated by any valid rollout that successfully negates the conclusion ($q_{\neg} = 1$):

$$r_{\text{hyp-only}} = \alpha + \gamma < \beta + \gamma \leq r_{\text{neg-Q}}.$$

This structurally precludes the "all hypotheses satisfied but conclusion asserted true" failure mode that affects naïve formulations (Li et al., 2026).

3.5 Verifier audit and operational measurements

The verifiers are produced by an LLM and therefore must be audited before they can support a reward signal. We hand-curate a suite of 47 unit tests, each pairing a verifier module with a small set of (*witness*, *expected conditions tuple*) cases covering at least one positive and one negative case per verifier. The production verifier corpus achieves a 47/47 pass rate on the audit suite.

Operationally, the sandbox is fast enough to support RLVR at our budget: across the held-out test set with each record's canonical witness fed through the oracle, the median wall-clock per call is 132 ms, p95 145 ms, and the maximum 326 ms. No test record approached the eight-second sandbox timeout.

Theorem-truth probe. We construct an auxiliary set of 347 *true* theorems drawn from the same theorem source. The model is asked to classify whether the provided theorem is true or false.

4 Training Setup

We train Qwen3-4B (Qwen Team, 2025) (`unsloth/Qwen3-4B`) using Supervised Fine-Tuning (SFT) followed by Group Relative Policy Optimization (GRPO) (Shao et al., 2024). SFT is performed over 3 epochs with a cosine learning rate schedule. For RLVR, we initialize from the SFT adapter and train for 4 epochs ($\approx$ 1,752 steps) using a KL penalty of 0.02 against the SFT reference policy. During RLVR, we sample $G = 4$ rollouts per prompt using vLLM (Kwon et al., 2023) to maintain high-throughput generation. Both training stages utilize LoRA (Hu et al., 2022) (rank 16, applied to all attention and MLP modules) and bfloat16 precision.

Domain	Strict	SFT	RLVR$_s$	Test
GroupTheory	874	366	434	74
RingTheory	565	237	280	48
NumberTheory	392	164	195	33
LinearAlgebra	233	98	115	20
FieldTheory	166	70	82	14
RealAnalysis	159	67	79	13
Strict-pool subtotals	**2389**	**1002**	**1185**	**202**

Final post-gating split sizes:
SFT = 1,000; Test = 201;
RLVR = 3,506 (strict-residual + 2,318 loose).

Table 1: Dataset statistics by domain. "Strict" = records whose teacher-generated canonical witness passes the verifier at generation time. The per-domain columns show the strict-pool subtotals: their sums (1,002 SFT and 202 test) are slightly larger than the final post-gating splits (1,000 and 201), reflecting 3 records dropped during oracle-gating. RLVR$_s$ is the strict-pool contribution to the RLVR prompt set; the remaining 2,318 RLVR prompts come from the loose pool (records with a valid verifier but no oracle-confirmed canonical witness at generation time). Total records produced by the pipeline: 4,707 (covering 3,485 unique theorems). Verifier audit: 47/47 unit tests pass; canonical-witness positive-reward rate: 0.98.

Notably, our entire pipeline is highly computationally accessible: training executes on a single NVIDIA RTX 5090 (32 GB) consumer GPU, with SFT completing in under five minutes and each RLVR run taking approximately nine hours. Full hyperparameters, optimizer configurations, and prompt templates are detailed in Appendix A.

5 Experimental Setup

5.1 Models

We report four model variants from this work: the unmodified **Qwen3-4B base**, **SFT** checkpoint, and two RLVR checkpoints **RLVR-dense** (Eq. 1 dense weights) and **RLVR-sparse** (sparse weights). For external context we evaluate five open-weights 7B math-specialised models, Abel-7B-002, DeepSeek-Math-7B-RL, Eurus-2-7B-PRIME, MAmmoTH2-7B-Plus and Qwen2.5-Math-7B-Instruct (Shao et al., 2024; Yang et al., 2024; Cui et al., 2025; Yue et al., 2023), and six frontier commercial APIs from the DeepSeek and Gemini families. Per-model breakdowns of the external comparators are deferred to Appendix C; in the main paper we sum-

marise them as a best-7B row and a frontier-API range.

5.2 Benchmarks

Counterexample (in-domain, n=201). Our held-out test split (§3.3), stratified by domain. The strict scoring criterion is $q_{\neg}{=}1 \wedge s_{\text{hyp}}{=}n_{\text{hyp}}$; we also report the relaxed *conclusion-pass* rate $q_{\neg}{=}1$.

Theorem-truth (n=347). The model is considered correct iff it recognises the theorem as true. We evaluate two prompt styles: *v2-counterexample* (the same prompt style as training; correct response is the `NONE` sentinel) and *truth-classify* (explicit "Answer: TRUE").

External math benchmarks. GSM8K (n=1319), MATH-500 (n=500), MMLU-college-math (n=100), AIME-2024 (n=30) (Cobbe et al., 2021; Hendrycks et al., 2021b,a). All four models use **identical** prompting (standard chain-of-thought); our SFT and RLVR splits contain **no** examples from these benchmarks and no training on their answer-extraction formats.

5.3 Decoding and metrics

We evaluate under two decoding regimes: *greedy* (temperature 0, $n{=}1$) and *sampled* (temperature 0.7, top-p 1.0, $n{=}8$). Pass@k for $k \in \{1, 3, 5, 8\}$ is computed with the unbiased estimator of Chen et al. (2021); greedy Pass@1 is reported as the headline number throughout. 95 % confidence intervals are percentile bootstrap intervals over problem ids with 10,000 resamples (the unbiased Pass@k estimator is applied per problem *before* resampling). Paired deltas (model A minus model B) bootstrap over the per-problem Pass@k differences. All scripts are released.

6 Results

6.1 Counterexample generation

Table 2 summarises the main result. Counterexample-only SFT lifts greedy Pass@1 modestly above the Qwen3-4B base (0.30 $\rightarrow$ 0.35; 95 % CI on the paired delta $[-0.030, +0.119]$), entirely consistent with no statistically significant gain at this sample size. Continuing with verifier-grounded RL, regardless of reward shape, yields a substantial further gain to 0.50 [0.43, 0.57] (paired delta versus the base $+0.199$ [+0.13, +0.27]).

Three points are worth emphasising. First, our 4B RLVR model exceeds every 7B math-specialist open-weights model we evaluated (full table in Appendix C); the best 7B baseline (Eurus-2-7B-PRIME at 0.30) is statistically indistinguishable from our untrained 4B base (paired CI $[-0.075, +0.065]$), which isolates the gain to the training method rather than parameter count. Second, the gain holds in expectation under sampled decoding: RLVR Pass@8 is 0.68, versus 0.57 for the best 7B baseline and 0.48 for the base. Third, our 4B RLVR model falls inside the range of greedy Pass@1 measured for six commercial frontier APIs (0.41–0.57); only one of the six APIs outperforms it. This supports the claim that a 4B model can be made competitive with frontier APIs on this task by training rather than by scale.

6.2 The imitation trap

Table 3 presents what we view as the central scientific finding of this work. Counterexample-only SFT collapses the model's ability to recognise true theorems. Under the counterexample-emit prompt format used throughout training, Pass@1 drops from 0.27 (base) to 0.00 (SFT) on all true theorems. The collapse is also visible, though less dramatic, under an explicit `Answer: TRUE/FALSE` classification prompt: 0.71 $\rightarrow$ 0.23. The base model retains genuine calibration; the supervised-only model does not.

Continuing from the same SFT adapter, verifier-grounded RL with a *sparse* outcome-only reward repairs and exceeds the base under both prompt styles (0.65 and 0.84 respectively). The dense reward only partially repairs the more demanding `v2_counterexample` probe (0.35). The dense and sparse variants are statistically indistinguishable on the in-distribution counterexample benchmark of §6.1 (Table 2) but differ by 30 percentage points on this probe.

This dissociation — equal in-distribution Pass@1, large out-of-distribution gap — is interpretable as evidence that the two reward shapes induce qualitatively different policies despite producing the same counterexample-generation performance. We give the mechanism in §6.3.

6.3 Why it happens: a reason-taxonomy analysis

We exploit the structured reward to label every test rollout with one of seven mutually exclusive *reasons* (Figure 1). Three observations explain the imitation trap.

SFT trades format errors for logical errors. The base model rarely emits the trained format at

Model	Pass@1 (greedy)	Pass@8 (sampled)	Format-OK	Parse-OK
Qwen3-4B base	0.30 [0.24, 0.37]	0.48 [0.41, 0.55]	0.45	0.42
+ SFT	0.35 [0.28, 0.41]	0.64 [0.57, 0.71]	**1.00**	0.82
+ RLVR (dense)	**0.50** [0.43, 0.57]	**0.68** [0.61, 0.74]	0.98	0.94
+ RLVR (sparse)	0.49 [0.42, 0.56]	0.67 [0.61, 0.74]	0.97	**0.95**
Best open-weights 7B (Eurus-2-7B-PRIME)	0.30 [0.24, 0.36]	0.57 [0.50, 0.64]	0.87	0.70
Frontier commercial APIs (range, $n_{\text{API}} = 6$)[a]	0.41 – 0.57	up to 0.77	—	—

Table 2: In-domain counterexample benchmark (n=201). Greedy Pass@1 is the headline metric; 95 % bootstrap CIs in brackets. [a]Range across six DeepSeek/Gemini frontier APIs. The full per-model open-weights table is in Appendix C.

Model	`v2_counterexample`	`truth_classify`
Qwen3-4B base	0.27 [0.22, 0.32]	0.71 [0.66, 0.76]
+ SFT	**0.00** [0.00, 0.00]	0.23 [0.18, 0.27]
+ RLVR (dense)	0.35 [0.29, 0.40]	0.76 [0.71, 0.80]
+ RLVR (sparse)	**0.65** [0.61, 0.70]	**0.84** [0.80, 0.88]

Table 3: Theorem-truth probe (n=347). Pass@1 (greedy) = fraction of theorems correctly recognised as true. `v2_counterexample`: correct response is the `NONE` sentinel. `truth_classify`: correct response is "`Answer: TRUE`".

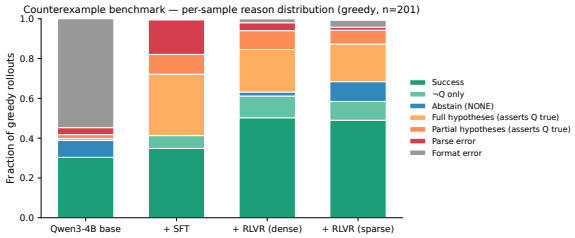

Figure 1: Per-sample reason distribution on the counterexample benchmark (greedy, n=201). Each rollout is labelled with one of seven reasons by the oracle: `success` ($q_\neg$=1 and all hypotheses satisfied), `neg_q_only` ($q_\neg$=1 but not all hypotheses), `none_on_false` (abstention on a false conjecture), `hyp_only` (all hypotheses satisfied but $q_\neg$=0 — the policy asserts the false conjecture is true), `partial`, `oracle_parse_error`, `no_tag`.

all: 55 % of its rollouts are `no_tag`. SFT solves this, but the corresponding mass does not move uniformly into `success`; instead, the `hyp_only` category — the policy satisfies the remaining hypotheses but asserts the (false) conclusion is true — rises from 1 % to 31 %. The supervised-only model learned the *form* of a counterexample without learning the calibration: it confidently emits objects that fail the negated-conclusion check.

Sparse RL re-learns abstention. The abstention rate (`none_on_false`) is 0 % under SFT, 2 % under dense RL, and 10 % under sparse RL. The sparse reward yields equal score for emitting a

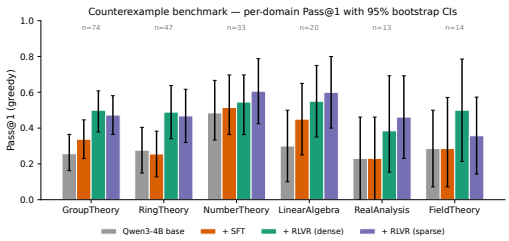

Figure 2: Per-domain counterexample Pass@1 (greedy) with 95 % bootstrap CIs. Annotations on top: number of test items per domain. The largest *absolute* gain is on LinearAlgebra (+0.30); the most striking result is on RealAnalysis, where RLVR-sparse gains +0.23 absolute despite the domain being only 6.7 % of the RLVR training distribution.

wrong witness and emitting `NONE`, so any rollout the policy is uncertain about is funnelled into abstention; under the dense reward, the partial-credit term $\alpha \cdot (s_{\text{hyp}}/n_{\text{hyp}})$ makes "try and partially succeed" strictly better than "abstain". This is the causal mechanism for the 30-percentage-point gap of Table 3.

Identical success rates hide diverging failure modes. While dense and sparse RL achieve nearly identical `success` fractions (0.50 vs 0.49), dense rewards push the remaining probability mass into over-eager `hyp_only` errors, whereas sparse rewards push it into `none_on_false` abstention.

We conclude that when a verifier yields a structured boolean tuple, an outcome-only aggregator generalizes better than partial credit. Rewarding the satisfaction of some, but not all, hypotheses actively trains over-eager counterexample emission, sacrificing true logical calibration.

6.4 Cross-domain consistency

Figure 2 stratifies the main result by domain. Two observations stand out. (i) Both RLVR variants improve on the base in every domain. There is no evidence of domain collapse or narrow specialisa-

Benchmark (n)	base	SFT	dense	sparse
GSM8K (1319)	0.83	0.79	0.88	**0.89**
MATH-500 (500)	0.13	0.54	0.63	**0.65**
MMLU-CM (100)	0.34	0.56	0.68	**0.69**
AIME-2024 (30)	0.00	0.07	**0.10**	0.07

Table 4: Out-of-domain transfer under standard chain-of-thought prompting (greedy, Pass@1). Identical prompt template across all four models; our SFT/RLVR pipeline contains no examples or answer-format training from these benchmarks.

tion. (ii) The largest relative gain comes from Real Analysis: base 0.23 → RLVR-sparse 0.46 (+0.23 absolute), despite Real Analysis being the smallest slice of the RLVR training set at 6.7 % (Table 1). This is the inverse of what an over-fitting model would do, and is consistent with the broader claim that the policy is learning a transferable reasoning skill rather than a domain-conditional shortcut.

Dense and sparse 95 % CIs overlap in almost every domain, supporting the in-distribution equivalence of §6.1 per-domain rather than as an aggregation artefact.

6.5 Out-of-domain transfer

Table 4 measures transfer to four standard mathematical-reasoning benchmarks. Since (i) all four models are evaluated under identical chain-of-thought prompting, and (ii) our SFT and RLVR data contain no GSM8K, MATH, MMLU or AIME examples and no training on their answer-extraction formats, the improvements are attributable to transfer rather than format adaptation in the training pipeline.

On every benchmark, training improves on the base; sparse and dense yield comparable gains, with sparse marginally ahead (except AIME-2024). While our trained variants do not learn this strict format from any in-pipeline data either, they apparently learn enough generic chain-of-thought structure during SFT and RLVR to score substantially higher. However, the GSM8K results isolate the true reasoning gain: on GSM8K, the base model already formats its answers correctly (scoring 0.83), yet sparse RLVR still provides a robust absolute improvement to 0.89 under the exact same prompt. Thus, while format adaptation contributes heavily to the magnitude of the MATH-500 jump, the underlying cross-benchmark shift reflects a genuine transfer of fundamental reasoning capabilities rather than mere stylistic compliance.

We interpret these results as supporting the broader narrative motivated in the introduction: counterexample-style supervision is not a narrow side-skill but a fundamental component of mathematical reasoning that conventional pre-training under-installs.

6.6 Qualitative analysis

Figure 3 grounds the imitation-trap claim in two concrete examples. The first is a true theorem on which the SFT model confidently produces a syntactically valid, well-formatted counterexample expression that none the less fails the negated-conclusion check (a `hyp_only` failure in the taxonomy of §6.3); the sparse RLVR model correctly emits the `NONE` sentinel; the dense model is intermediate. The second example illustrates the symmetric failure mode of sparse: a false conjecture on which the dense model produces a clean witness while the sparse model abstains incorrectly. We discuss the trade-off in §7.

7 Discussion

Why sparse generalises better on calibration. The mechanism in §6.3 is interpretable. Under the dense reward, the partial-credit term $\alpha \cdot (s_{\text{hyp}}/n_{\text{hyp}})$ guarantees a positive reward whenever the policy satisfies at least one remaining hypothesis, even when it asserts the (false) conclusion is true. This actively trains the policy to *always emit a counterexample expression*: doing so on a false conjecture yields up to $\alpha+\gamma = 0.35$, much better than the certain -1 of `none_on_false`. Sparse rewards remove this incentive: the policy is rewarded only for correctly negating the conclusion, and abstention is no worse than a wrong witness. The behavioural consequence is the higher `none_on_false` rate on the in-distribution test set (10 % vs 2 %) and, crucially, the preservation of abstention as a viable action when the policy is shown a *true* theorem at test time. The dense reward does not preserve this; the partial-credit term is in tension with calibration.

We conjecture this generalises to other settings where (i) the reward signal already returns a structured boolean tuple, and (ii) the failure mode of interest is over-eager output rather than under-formatting. Whether this conjecture survives at scale is left to future work.

Deterministic verification beats LLM-as-judge where the verification problem is decidable. The verifier substrate is reproducible, fast (median

Example 1 (TRUE theorem from the theorem-truth probe).
Continuous convergence theorem (Real Analysis). The theorem states that if a sequence of continuous functions satisfies the required convergence assumptions, then the limit function is continuous.
SFT output: Counterexample: $X = [0, 1]$, $f_n(x) = x^n$, and

$$f(x) = \begin{cases} 0 & x < 1, \\ 1 & x = 1. \end{cases}$$

The SFT model confidently proposes the classical sequence $f_n(x) = x^n$ as a counterexample, despite the theorem actually being true and the construction violating the required convergence assumptions. RLVR-dense becomes substantially less assertive and ultimately abstains (`NONE`), while RLVR-sparse correctly identifies the theorem as true and emits `NONE`.

Example 2 (FALSE conjecture from the counterexample benchmark).
Minimal polynomial of $\sqrt{a} + \sqrt{b}$ (Field Theory). The original theorem requires that a, b, and ab are all non-squares in the base field. The benchmark removes the assumption that a is not a square.
RLVR-dense witness:

$$(F, a, b) = (\mathbb{Q}, 1, 2).$$

Since 1 is a square in $\mathbb{Q}$, the claimed quartic polynomial is no longer the minimal polynomial of $1 + \sqrt{2}$:

$$x^2 - 2x - 1 \neq x^4 - 6x^2 + 1.$$

RLVR-dense successfully identifies this compact counterexample, whereas RLVR-sparse abstains and emits `NONE`, illustrating the trade-off introduced by sparse rewards: improved calibration on true theorems at the cost of increased false abstentions on genuinely false conjectures.

Figure 3: Two qualitative examples illustrating the imitation trap (top) and a sparse-reward failure mode (bottom).

132 ms), audited (47/47), and immune to judge-model drift. Its scope is bounded by what is decidable in Python with SymPy/Z3; for the algebraic subdomains in this paper this is sufficient. We note that COUNTERMATH (Li et al., 2025), the closest comparable counterexample benchmark, uses an LLM-judge protocol; while this admits a broader domain coverage, it carries judge-model dependence that complicates reproducibility. The two approaches are complementary rather than substitutable.

Counterexample reasoning as a fundamental capability. The cross-domain consistency of §6.4 and the out-of-domain transfer of §6.5 together support a stronger reading of the contribution: training on structured falsification does not just produce a model that disproves better, it produces a model that reasons better in general. The Real Analysis result (6.7 % training share, +0.23 absolute gain) and the +52 percentage-point gain on MATH-500 under unchanged prompting are the strongest evidence for this reading. The literature on the "hallucination tax" (Song et al., 2025) predicts that standard imitation-only training degrades calibration; the present results suggest that verifier-grounded RL with an outcome-only reward not only avoids that pathology but installs a transferable inverse-reasoning skill.

8 Conclusion

We have presented a deterministic, large-scale dataset and verification environment for counterexample generation on undergraduate algebra and real analysis, and used it to identify and explain an *imitation trap*: supervised fine-tuning on counterexample-only data collapses the policy's ability to recognise true theorems, and verifier-grounded reinforcement learning with a sparse outcome-only reward repairs and exceeds the original calibration where a partial-credit dense reward does not. A Qwen3-4B model trained this way exceeds every 7B open-weights math specialist we evaluated on the in-domain benchmark, remains competitive with frontier commercial APIs, and transfers under unchanged prompting to GSM8K, MATH-500 and MMLU-college-math — evidence that counterexample reasoning is a fundamental, under-installed skill rather than a narrow specialisation. We release the dataset, the verifier modules and audit suite, the witness-schema generator, and the full evaluation pipeline, and we view extension to topology and higher-order logic, multi-seed validation, and integration with a Lean back-end as the natural next steps.

Limitations

We discuss several limitations of the present work.

Single seed. All headline results are produced from a single training seed (42), held constant across SFT, both RLVR variants, and every evaluation run. This uniformity is a deliberate methodological choice rather than a convenience: by fixing the seed across configurations, we avoid the per-configuration selection bias that would arise from reporting the best of several runs, and the relative comparisons between SFT, dense RL, and sparse RL therefore reflect differences induced by the training method rather than post-hoc seed selection. We further report 95 % bootstrap confidence intervals (§5.3, 10,000 paired resamples over problem ids), which quantify test-set variance and, through paired deltas between checkpoints, whether observed gaps would survive any subset of the test set. What these intervals do not characterise is variance over training seeds, and we acknowledge that direct replication remains the proper test of robustness; we note that the central comparison of the paper, the 30-point sparse-vs-dense gap on the calibration probe (Table 3), is substantially larger than the typical paired-bootstrap width of our in-distribution comparisons, which provides indirect evidence that the dissociation is unlikely to vanish under reseeding, though not a substitute for direct replication. Replicate runs with two additional seeds for both RLVR variants are planned for the camera-ready / rebuttal phase.

Verifier scope. Our verifier substrate is Python + SymPy + Z3; this is sufficient for the six algebraic and real-analysis subdomains we evaluate, but is not sufficient for topology, general higher-order logic, or claims whose verification reduces to undecidable fragments. Extending the verifier substrate (for example via a Lean backend, or with continuous-FOL extensions for metric spaces) is a natural follow-up.

Audit size. The verifier audit suite (§3.5) is 47 hand-curated tests. A larger audit suite would strengthen confidence in the reward signal; we view this as the most pressing engineering follow-up. Larger audit set with at least 80 hand-curated tests is planned for the camera-ready / rebuttal phase.

Mutation strategy. We mutate only by dropping a single hypothesis. Conclusion-mutation, multi-hypothesis dropping, and AST-level mutation (Wu et al., 2024) are unexplored; they may produce qualitatively different distributions of false conjectures and would change the per-domain breakdown.

No COUNTERMATH cross-evaluation. Cross-evaluation on COUNTERMATH (Li et al., 2025) would directly compare our trained models against the published frontier on a counterexample benchmark with broader domain coverage. We were unable to complete this comparison in time: the authors do not publicly release the test labels or full evaluation scripts; predicted outputs must be submitted for LLM-judged scoring on the authors' side. We leave this for future work.

Single base-model scale. All trained variants use Qwen3-4B as the base. We do not test whether the imitation-trap finding survives at larger scales. The mechanism (§6.3) is interpretable enough that we conjecture it does, but this remains conjecture.

Teacher model dependence. The verifiers used to evaluate the validity of the counter-examples are generated by a single teacher model (Gemini Pro 3.1). Although additional filtering was made and the results are positive, the pipeline retains a dependence on the teacher's output quality.

Ethics and Risk Considerations

The work studies counterexample generation for undergraduate-level algebraic and real-analysis theorems, using LLM-generated Python verifiers and a hypothesis-dropping mutation pipeline. The artifacts produced (a dataset of false mathematical conjectures, per-theorem verifier modules, and a small (4B) trained model specialized for falsifying such conjectures) do not present obvious dual-use risks beyond those general to research on LLM mathematical reasoning. The dataset contains no human-derived data, no personal information, and no naturally occurring text from which PII could be derived; the trained models are not deployed in user-facing settings; and the methodology does not enable capabilities that meaningfully extend beyond existing open-source mathematical reasoning models.

References

Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde de Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, and 1 others. 2021. Evaluating large language models trained on code. *arXiv preprint arXiv:2107.03374*.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias

Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, and 1 others. 2021. Training verifiers to solve math word problems.

Ganqu Cui, Lifan Yuan, Zefan Wang, Hanbin Wang, Yuchen Zhang, Wendi Li, Bingxiang He, Yuchen Fan, Tianyu Yu, Qixin Xu, Weize Chen, Jiarui Yuan, Huayu Chen, Kaiyan Zhang, Xingtai Lv, Shuo Wang, Yuan Yao, Xu Han, Hao Peng, and 5 others. 2025. Process reinforcement through implicit rewards. *arXiv preprint arXiv:2502.01456*.

Leonardo De Moura and Nikolaj Bjørner. 2008. Z3: An efficient smt solver. In *Tools and Algorithms for the Construction and Analysis of Systems: 14th International Conference, TACAS 2008*, pages 337–340. Springer.

Daya Guo and 1 others. 2025. Deepseek-r1 incentivizes reasoning in llms through reinforcement learning. *Nature*, 645(8081):633–638.

Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. 2021a. Measuring massive multitask language understanding.

Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. 2021b. Measuring mathematical problem solving with the math dataset.

Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. 2022. LoRA: Low-rank adaptation of large language models. In *International Conference on Learning Representations*.

Ryo Kamoi, Yusen Zhang, Nan Zhang, Sarkar Snigdha Sarathi Das, Ranran Haoran Zhang, Wenpeng Yin, and Rui Zhang. 2026. Efficient prm training data synthesis via formal verification. In *Findings of the Association for Computational Linguistics: ACL 2026*.

Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph E. Gonzalez, Hao Zhang, and Ion Stoica. 2023. Efficient memory management for large language model serving with PagedAttention. In *Proceedings of the 29th Symposium on Operating Systems Principles*.

Yinghui Li, Jiayi Kuang, Haojing Huang, Zhikun Xu, Xinnian Liang, Yi Yu, Wenlian Lu, Yangning Li, Xiaoyu Tan, Chao Qu, Ying Shen, Hai-Tao Zheng, and Philip S. Yu. 2025. One example shown, many concepts known! counterexample-driven conceptual reasoning in mathematical llms.

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

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

Aaron Meurer, Christopher P Smith, Mateusz Paprocki, Ondřej Čertík, Sergey B Kirpichev, Matthew Rocklin, AMiT Kumar, Sergiu Ivanov, Jason K Moore, Sartaj Singh, and 1 others. 2017. Sympy: symbolic computing in python. *PeerJ Computer Science*, 3:e103.

Jialin Ouyang. 2025. Treecut: A synthetic unanswerable math word problem dataset for llm hallucination evaluation. *arXiv preprint arXiv:2502.13442*.

Qwen Team. 2025. Qwen3 technical report. *Preprint*, arXiv:2505.09388.

Rulin Shao, Shuyue Li, Rui Xin, Scott Geng, Yiping Wang, Sewoong Oh, Simon Du, Nathan Lambert, Sewon Min, Ranjay Krishna, Yulia Tsvetkov, Hannaneh Hajishirzi, Pang Koh, and Luke Zettlemoyer. 2025. Spurious rewards: Rethinking training signals in rlvr.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haoran Zhang, Mingchuan Zhang, YK Li, Y Wu, and 1 others. 2024. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300*.

Shiven Sinha, Shashwat Goel, Ponnurangam Kumaraguru, Jonas Geiping, Matthias Bethge, and Ameya Prabhu. 2025. Can language models falsify? evaluating algorithmic reasoning with counterexample creation. *Preprint*, arXiv:2502.19414.

Linxin Song, Taiwei Shi, and Jieyu Zhao. 2025. The hallucination tax of reinforcement finetuning. *arXiv preprint arXiv:2505.13988*.

Peiyi Wang, Lei Lei, Ruiyang Wu, and 1 others. 2023. Math-shepherd: Verify and reinforce llms step-by-step in mathematical reasoning. *arXiv preprint arXiv:2312.08935*.

Shaonan Wu, Shuai Lu, Yeyun Gong, Nan Duan, and Ping Wei. 2024. Alchemy: Amplifying theorem-proving capability through symbolic mutation. *arXiv preprint arXiv:2410.15748*.

An Yang, Beichen Zhang, Binyuan Hui, Bofei Gao, Bowen Yu, Chengpeng Li, Dayiheng Liu, Jianhong Tu, Jingren Zhou, Junyang Lin, and 1 others. 2024. Qwen2.5-math technical report: Toward mathematical expert model via self-improvement. *arXiv preprint arXiv:2409.12122*.

Lifan Yuan, Ganqu Cui, Hanbin Wang, Ning Ding, and 1 others. 2024. Eurus: A suite of open-source llms for math and code. *arXiv preprint arXiv:2404.02078*.

Xiang Yue, Xingwei Ni, Kai Zhang, Tianyu Zheng, Liu Ruoqi, Ge Zhang, Samuel Stevens, Dongfu Jiang, Weiming Ren, Minghuan Sun, and 1 others. 2023. Mammoth: Evaluating large language models on mixed math data. *arXiv preprint arXiv:2309.05653*.

A Full Hyperparameters

We list the exact hyperparameters used for the SFT and RLVR runs reported in this paper. The configs will be released verbatim in `configs/full_run_dense.yaml` (SFT and dense-RLVR) and `configs/full_run_sparse.yaml` (sparse-RLVR).

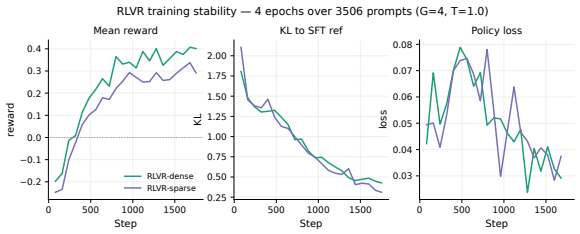

Figure 4: Per-step training curves for the dense and sparse RLVR runs ($\approx$ 1,752 steps over four epochs of 3,506 prompts). Mean reward rises monotonically and crosses zero by step $\sim$240 for both variants; KL to the SFT reference decays smoothly; loss is bounded and oscillatory but stable.

Figure 4 shows the per-step training curves for both RLVR runs. The reward trajectory rises from approximately -0.20 at step 80 to $+0.40$ (dense) / $+0.30$ (sparse) at convergence; the KL penalty to the SFT reference declines from ~ 2.0 to ~ 0.5 as the policy explores away from imitation; the policy loss is bounded and oscillatory but does not diverge. Both runs complete the full four epochs without any restart, gradient explosion or reward collapse.

B Verifier Audit Suite

The audit suite contains 47 hand-curated unit tests, each pairing a verifier module with a witness expression and an expected outcome. Forty-one tests are *positive* (the witness is expected to satisfy all conditions, i.e. `conditions(witness) == [True, . . . , True]`); six are *targeted negative* tests that probe specific failure modes (e.g. a hypothesis that the verifier must distinguish, or a witness that satisfies the remaining hypotheses but does *not* negate the conclusion). Per-domain coverage and pass rates are reported in Table 6.

C Per-Model Baseline Breakdowns

Table 7 reports the per-model counterexample-benchmark scores for the five 7B open-weights math specialists summarised in the body. Table 8 reports the same for the six commercial-API frontier models that contribute to the "frontier API range" band in Table 2. We additionally report the Qwen2.5-Math-1.5B base in Table 7 as a smaller comparison point.

Supervised fine-tuning (SFT)	
Base model	`unsloth/Qwen3-4B`
Chat template	`qwen-3`
Max sequence length	2048
Precision	bfloat16
LoRA rank r / scale α	16 / 16
LoRA dropout	0
LoRA targets	all attn. + MLP
Gradient checkpointing	unsloth
Per-device batch	16
Gradient accumulation	2
Effective batch	32
Epochs	3
Learning rate	2×10^{-4}
LR schedule / warmup	cosine / 0.05
Optimiser	AdamW
Weight decay	0.0
Max grad norm	1.0
Eval-loss split fraction	0.05
Seed	42
RLVR (GRPO from SFT adapter)	
Algorithm	TRL `GRPOTrainer`
Rollout backend	vLLM ($gpu_util = 0.5$)
Max LoRA rank	16
G (generations / prompt)	4
Max prompt / completion tokens	768 / 1024
Temperature	1.0
Top-p	1.0
Per-device batch	4
Gradient accumulation	8
Effective batch	32
Epochs	4
Total steps	$\approx$ 1,752
Learning rate	5×10^{-6}
LR schedule / warmup	constant / 0.1
PPO clip ε	0.2
KL coefficient β_{KL}	0.02
Max grad norm	0.5
Oracle timeout	8 s
Reward-pool workers	4
Seed	42
Oracle reward weights (Eq. 1)	
Dense: $(\alpha, \beta, \gamma, \delta, \varepsilon)$	$(0.25, 0.65, 0.10, 0.30, 1.00)$
Sparse: $(\alpha, \beta, \gamma, \delta, \varepsilon)$	$(0,\ 1,\ 0,\ 0,\ 1)$

Table 5: Hyperparameters used for SFT and both RLVR variants. SFT wall-clock on a single RTX 5090 32 GB is under five minutes; each RLVR run is approximately nine hours.

D Witness-Schema Generator and Example Schemas

The witness-schema stage (§3.2) is implemented in `tools/inject_witness_schema.py`. The tool iterates over verifier modules and queries a teacher model (DeepSeek via the OpenAI-compatible API client) with the verifier source code and the canoni-

Domain	#tests	#passed	rate
GroupTheory	12	12	1.00
RealAnalysis	10	10	1.00
LinearAlgebra	9	9	1.00
RingTheory	9	9	1.00
FieldTheory	5	5	1.00
NumberTheory	2	2	1.00
Total	**47**	**47**	**1.00**

Table 6: Verifier-audit suite. The full suite will be released at `tests/oracle/verifier_unit_tests.jsonl`. Number-Theory coverage is the smallest because the corpus contains comparatively few NumberTheory verifiers that we hand-audited; expanding coverage in this subdomain is one of the engineering follow-ups noted in the Limitations.

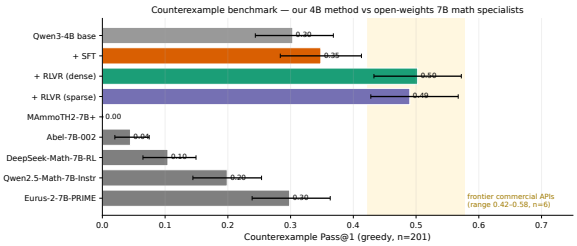

Figure 5: Counterexample Pass@1 (greedy) for all comparators evaluated. Our trained 4B models are in colour; open-weights 7B specialists are in grey; the shaded yellow band shows the range across six frontier commercial APIs.

cal witness expression, asking for a single-sentence Python-shape description of an acceptable witness, with a concrete example.

Representative schemas (one per subdomain).

> **GroupTheory.** "Python tuple (G, H) where both G and H are `sympy.combinatorics.PermutationGroup` objects constructed via SymPy group constructors (e.g., `SymmetricGroup(n)`, `AlternatingGroup(n)`, `DihedralGroup(n)`, `CyclicGroup(n)`). Example syntax: `(SymmetricGroup(3), DihedralGroup(4))`."

> **RingTheory.** "An `RModule` instance with attributes: `ring` (`CommutativeRing`) and `generators_count` (`int`). Example: `RModule(Z, 2)`."

> **FieldTheory.** "A SymPy `Domain` object representing an integral domain that is not a field. Example: `ZZ` from SymPy."

> **LinearAlgebra.** "A SymPy `Matrix` object (square, non-real). Example: `Matrix([[0, 1], [-1, 0]])`."

> **RealAnalysis.** "Python tuple (f, a, b, p) where f is a SymPy expression (or string) in terms of `Symbol('x')`, a and b are real numbers $(a < b)$, p is a real number."

> **NumberTheory.** "A Python `int` or `sympy.Integer` (single integer). Example: `5`."

The schemas are short (10–40 tokens) and add a small overhead to the user prompt (median schema length is well below the SFT user-prompt median of 120 tokens reported in the dataset stats).

E Eval-Suite Configuration and Reproducibility

All evaluation in the main paper is orchestrated by the multi-model multi-benchmark suite runner `python -m training.eval.run_suite –config configs/eval/thesis_full.yaml`, which iterates the cartesian product {model} × {benchmark} × {prompt-style} × {greedy, sampled} and writes one JSON per cell. Each per-cell JSON contains a `summary` block (Pass@k, by-domain, and (where applicable) subset metrics) and a `per_record` block (one record per test problem with all sample-level fields: `completion`, `extracted`, `correct`, `reason`, `format_ok`, `parse_ok`, `is_none`, oracle timing, etc.).

Reproducing the paper from the released artefacts.

1. **Bootstrap CIs.** `python -m paper_preparation.scripts.bootstrap_cis` regenerates `bootstrap_cis.csv` and `bootstrap_deltas.csv`.
2. **Reason distribution (Fig. 1).** `python -m paper_preparation.scripts.eval_reason_dist` regenerates the reason-counts table.
3. **All figures.** `python -m paper_preparation.scripts.make_plots –all` regenerates Figures 1, 2, 4, and 5 from the released CSVs.

Dataset fingerprints. Each suite run writes a `dataset_fingerprint.json` file containing the SHA-256 hash and line count of every input JSONL used (the test set, the theorem-truth probe, and the external-benchmark sources). These fingerprints are sufficient to verify that all numbers in this paper were produced from the released splits.

Seed. A single seed (42) was used for the SFT and both RLVR runs. The configs above are sufficient to reproduce all training runs deterministically up to GPU non-determinism in the rollout

Model	Pass@1 (greedy)	Pass@1 (sampled)	Pass@3 (sampled)	Pass@8 (sampled)
Eurus-2-7B-PRIME	0.299 [0.24, 0.36]	0.292 [0.24, 0.34]	0.458 [0.40, 0.52]	0.567 [0.50, 0.64]
Qwen2.5-Math-7B-Instruct	0.199 [0.14, 0.25]	0.063 [0.05, 0.08]	0.155 [0.12, 0.19]	0.289 [0.23, 0.35]
DeepSeek-Math-7B-RL	0.105 [0.06, 0.15]	0.072 [0.05, 0.09]	0.171 [0.13, 0.21]	0.308 [0.25, 0.37]
Abel-7B-002	0.045 [0.02, 0.07]	0.052 [0.04, 0.07]	0.124 [0.09, 0.16]	0.224 [0.17, 0.28]
MAmmoTH2-7B-Plus	0.000 [0.00, 0.00]	0.040 [0.03, 0.05]	0.106 [0.08, 0.14]	0.219 [0.16, 0.28]
Qwen2.5-Math-1.5B (smaller ref.)	0.010 [0.00, 0.02]	0.017 [0.01, 0.03]	0.045 [0.03, 0.07]	0.095 [0.05, 0.13]

Table 7: Open-weights baselines on the counterexample benchmark (n=201). All numbers Pass@k with 95 % bootstrap CIs over problem ids. Sampled decoding uses $n = 8$, temperature 0.7. Source: `paper_preparation/results/bootstrap_cis.csv`.

Model	Pass@1 (greedy)	Pass@1 (sampled)	Pass@3 (sampled)	Pass@8 (sampled)
Gemini-3.1-Flash-Lite	**0.577** [0.51, 0.64]	**0.565** [0.50, 0.63]	0.638 [0.57, 0.70]	0.687 [0.62, 0.75]
Gemini-3.5-Flash	0.493 [0.42, 0.56]	—	—	—
Gemini-3-Flash-Preview	0.468 [0.40, 0.54]	0.489 [0.44, 0.53]	**0.708** [0.65, 0.76]	**0.771** [0.71, 0.83]
DeepSeek-Pro	0.428 [0.36, 0.49]	0.437 [0.38, 0.49]	0.551 [0.49, 0.61]	0.622 [0.55, 0.69]
Gemini-3.1-Pro	0.428 [0.36, 0.50]	—	—	—
DeepSeek-Flash	0.423 [0.35, 0.49]	0.432 [0.38, 0.49]	0.562 [0.50, 0.62]	0.637 [0.57, 0.70]

Table 8: Frontier commercial APIs on the counterexample benchmark (n=201). Dashes indicate sampled passes that were not run because token-budget constraints prevented an 8-sample run within the budget. Source: `paper_preparation/results/bootstrap_cis.csv`.

backend (vLLM tensor-parallel sampling); the reported test-set numbers are robust to this residual non-determinism within the reported 95 % bootstrap CIs.