Improving Lean4 Autoformalization via Cycle Consistency Fine-tuning

Stanford CS224N Custom Project

Arsen Shebzukhov
Department of Computer Science
Stanford University
arsene@stanford.edu

Abstract

Autoformalization - automatically translating natural language mathematical texts into formal proof language such as Lean4 - can help accelerate AI-assisted mathematical research, be it via proof verification or proof search. I fine-tune Qwen3.5-2B with LoRA for natural language to Lean4 formalization on FineLeanCorpus and consider three training regimes: supervised fine-tuning (SFT) with curriculum learning (difficulty 1→10), SFT without curriculum ordering, and reinforcement learning using group relative policy optimization (GRPO) with a cycle consistency reward. Cycle consistency measures how well the meaning of a statement is preserved through a NL $\rightarrow$ Lean4 $\rightarrow$ NL$'$ loop, computed as cosine similarity of off-the-shelf sentence embeddings. On an unseen subset of FineLeanCorpus (FLC) and on PutnamBench, RL substantially outperforms both SFT variants (mean cycle consistency 0.669 vs. 0.513 on FLC; 0.561 vs. 0.422 on PutnamBench), while increasing cross-entropy loss by only 0.011 nats, with minimal impact on formalization quality. Curriculum ordering provides no measurable benefit over shuffled training.

1 Key Information to include

- Mentor: Luke Bailey
- External Collaborators (if you have any): None
- Sharing project: None
- Late Days (if you are using any):

Team member	Late days remaining
Arsen Shebzukhov	6
Total pooled	6
Team size	1
Late days applied for the report	3

2 Introduction

Formal proof assistants such as Lean4 allow mathematical statements and proofs to be verified by checking whether their formal encoding compiles correctly in a special functional programming language. One of the challenges of this process is transcribing the natural mathematical language (NL) into formal Lean4 statements, that can be fed to the compiler. *Autoformalization* is the automatization of this process. Despite substantial progress in large language model (LLM) reasoning, autoformalization remains difficult: models frequently produce syntactically plausible but semantically incorrect Lean4 code that compiles yet does not capture the intended mathematical meaning.

In this work I approach autoformalization as a sequence-to-sequence problem and explore how different training strategies affect the quality of the NL→Lean4 translation. Motivated by cycle-consistency of generative models [1], I define a *cycle consistency* metric: a good formalization $\hat{l} = f(s)$ of a statement s should allow a back-translator g to recover the original meaning, i.e. $g(f(s))$ is similar to s. I use cosine similarity of sentence embeddings from an off-the-shelf model as evaluation signal and reward for RL.

My contributions are:

1. A fine-tuning pipeline for Lean4 autoformalization using LoRA on Qwen3.5-2B, with PutnamBench decontamination of training data.
2. An ablation study of difficulty-ordered SFT against SFT on total shuffled dataset, showing no significant benefit for the chosen scale (both in terms of model parameters and dataset).
3. A GRPO-based RL fine-tuning using cycle consistency as a reward, which shows a 0.156 improvement in mean cycle consistency over the SFT baseline on held-out FLC data, with minimal cross-entropy deterioration.

3 Related Work

Autoformalization. Early work on autoformalization used retrieval and rule-based methods, whereas the approach using LLM has recently gained popularity driven by the increase in the number of openly available large corpora of NL/Lean4 pairs. FineLeanCorpus [2] provides 509k such pairs with difficulty ratings and domain labels, which I used as training data. PutnamBench [3] provides 660 competition-level problems kept as a held-out evaluation benchmark. The CriticLean pipeline [2] improves formalization yield by combining a Lean4 compiler feedback loop with a critic model that validates semantic correctness; my RL approach is complementary, optimising for semantic round-trip fidelity rather than compilation.

Cycle consistency in NLP. Cycle consistency was popularised by CycleGAN [1] for unpaired image translation. In NLP, back-translation [4] exploits a similar idea for low-resource machine translation. Dreano et al. [5] apply a discriminator-less CycleGAN to NMT, showing that retro-translation should recover source sentences. Most directly related, [6] use cycle consistency as an implicit quality signal for LLM translation, showing that larger models achieve higher consistency - motivating its use as a training signal. Cycle-Instruct [7] applies dual self-training with cycle consistency for instruction tuning without human seeds.

RL for mathematical reasoning. GRPO [8] is a policy-gradient algorithm that avoids a separate critic model by normalising rewards within groups of sampled completions. It has been applied to mathematical reasoning with verifiable rewards (compiler or symbolic checker). I adopt GRPO with a soft (embedding-based) reward, which avoids the latency cost of a Lean4 REPL while still providing a semantic signal.

Parameter-efficient fine-tuning. I use LoRA [9] throughout, restricting updates to the feed-forward sublayers of the transformer, which keeps memory and compute costs manageable on a single GPU.

4 Approach

4.1 Problem Formulation

Let $\mathcal{S}$ denote the space of natural language mathematical statements and $\mathcal{L}$ the space of Lean4 theorem declarations. Two models are trained:

- $f_\theta : \mathcal{S} \to \mathcal{L}$ - the **nl2lean** model (trainable).
- $g_\phi : \mathcal{L} \to \mathcal{S}$ - the **lean2nl** back-translator (frozen after SFT).

The *cycle consistency score* of a statement $s \in \mathcal{S}$ under model f_θ is:

$$\mathcal{C}_\theta(s) = \cos\big(\mathbf{e}(s),\ \mathbf{e}(g_\phi(f_\theta(s)))\big), \tag{1}$$

where $\mathbf{e}(\cdot)$ denotes the sentence embedding produced by `all-MiniLM-L6-v2`. This quantity serves as both the GRPO reward during RL training and the primary evaluation metric.

4.2 Model and LoRA Configuration

I use Qwen3.5-2B [10] as the base model, to which I apply LoRA adapters [9] restricted to the feed-forward network (FFN) sublayers only (`gate_proj`, `up_proj`, `down_proj`), with rank $r = 16$ and scaling $\alpha = 32$, resulting in approximately 9.4M trainable parameters out of 2B total. Attention layers are excluded because Qwen3.5 uses a hybrid linear/full attention architecture with non-standard module naming. All inputs are truncated to 512 tokens (covering >99% of the dataset). Training uses `bfloat16` precision throughout.

Baseline. The base Qwen3.5-2B with 3-shot prompting serves as the baseline, establishing that the pretrained model cannot formalize without fine-tuning (mean $\bar{\mathcal{C}} = 0.093$ on FLC, 0.034 on PutnamBench). I also use the Qwen3.5-9B model for comparison, with the same setup and dataset, restricted to SFT curriculum training only due to lack of compute.

4.3 SFT: Curriculum and No-Curriculum

Both SFT variants use the same data: FineLeanCorpus filtered to remove duplicates, compile errors, and examples with TF-IDF cosine similarity > 0.5 to any PutnamBench problem. Examples are capped at 10k per difficulty level, giving a dataset of 74.7k training and 3.93k validation examples across difficulties 1 to 10. There are significantly fewer examples for difficulties 8, 9, and 10. The dataset was uploaded as `arc-cola/flc-n2l-2b`.

The *curriculum* variant trains sequentially on difficulty $d = 1, 2, \ldots, 10$, running one epoch per difficulty and carrying the LoRA adapter forward between stages. The *no-curriculum* variant (ablation) trains on the identical data in a single shuffled pass. Both use AdamW with learning rate 2×10^{-4}, cosine scheduler, warmup for 3% of each stage, effective batch size 32, and cross-entropy loss.

4.4 RL via GRPO

The RL phase fine-tunes the nl2lean model (initialised from the SFT curriculum checkpoint) using GRPO [8]. The lean2nl back-translator is frozen throughout.

Dataset. I construct an RL prompt dataset by taking FineLeanCorpus minus all SFT training and validation examples, then stratifying to at most 1k prompts per difficulty level (difficulties 1 to 7, the range where held-out examples are available), giving 7.7k prompts total, uploaded as `arc-cola/flc-n2l-rl`.

Reward. For each generated Lean4 completion $\hat{l}$ in response to prompt s, the reward is:

$$r(\hat{l}, s) = \mathcal{C}_\theta(\hat{l}, s) = \cos\big(\mathbf{e}(s),\ \mathbf{e}(g_\phi(\hat{l}))\big). \tag{2}$$

Back-translation uses greedy decoding with the frozen lean2nl model and sentence embeddings are computed on CPU with `all-MiniLM-L6-v2`.

GRPO update. For each prompt s, GRPO samples $G = 8$ completions $\{o_i\}_{i=1}^{G}$ from f_θ and computes rewards $\{r_i\}$. Within-group reward normalisation gives advantage estimates:

$$\hat{A}_i = \frac{r_i - \mu_r}{\sigma_r + \varepsilon}. \tag{3}$$

The policy is updated to maximise the clipped surrogate objective while a KL penalty term $\beta \cdot D_{\mathrm{KL}}(f_\theta \| f_{\theta_{\mathrm{ref}}})$ keeps the model close to the SFT curriculum reference policy $f_{\theta_{\mathrm{ref}}}$. I use learning rate 10^{-5}, batch size 1, gradient accumulation 16 and the default β of 0.04.

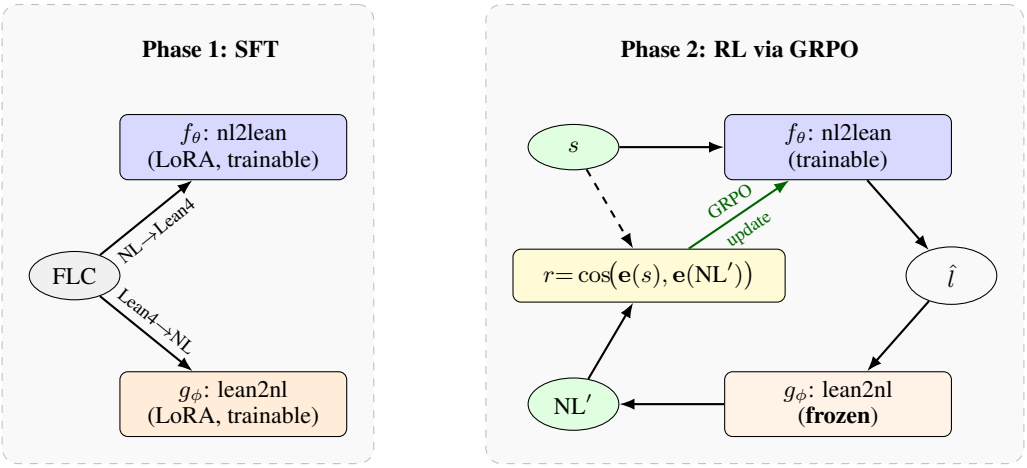


Figure 1: Training pipeline. **Phase 1:** f_θ and g_ϕ are fine-tuned independently via SFT on FineLean-Corpus, on the NL→Lean4 and Lean4→NL directions respectively. **Phase 2:** f_θ is further fine-tuned via GRPO with g_ϕ frozen. The reward is the cosine similarity between the original statement s and the back-translated NL′; both live in the same natural language space $\mathcal{S}$.

5 Experiments

5.1 Data

FineLeanCorpus (FLC). I use `m-a-p/FineLeanCorpus` [2], a dataset of 509k NL/Lean4 pairs annotated with difficulty scores (1 to 10) and mathematical domain labels. After deduplication and removal of examples with Lean4 compiler errors, I perform TF-IDF-based decontamination against PutnamBench (removing FLC examples with cosine similarity > 0.5 to any PutnamBench problem, top-4 per problem).

The resulting data is split into: SFT train/val (`arc-cola/flc-n2l-2b`), RL prompts (`arc-cola/flc-n2l-rl`), and a held-out evaluation set (`arc-cola/flc-final-eval`) of 700 examples (100 per difficulty, difficulties 1 to 7). All of the difficulty 8 to 10 examples were used for training data and are therefore missing from the final evaluation set.

PutnamBench. `lmms-lab/PutnamBench` [3] contains 660 Putnam competition problems with human-written Lean4 theorem declarations. It covers the hard end of the difficulty distribution missing in the FLC final evaluation set.

5.2 Evaluation Metric

The primary metric is *mean cycle consistency* $\bar{\mathcal{C}}$ (Equation 1), computed with greedy decoding and sentence embeddings from `all-MiniLM-L6-v2`. As a secondary metric I report cross-entropy loss on the FLC validation split, computed with teacher forcing to assess whether RL degrades the model's basic language modeling ability. I also report Mann-Whitney U tests to assess statistical significance of cross-model differences.

5.3 Experimental Details

All 2B experiments run on a single NVIDIA L40S (46GB). The 9B curriculum SFT comparison runs on an A100 SXM4 80GB (vast.ai). RL training uses two L40S GPUs: nl2lean on `cuda:0`, lean2nl frozen on `cuda:1`.

The RL run was interrupted at optimizer step 100 and resumed from checkpoint; a transient reward drop is visible at the resume point (Figure 8), consistent with optimizer state reconstruction, and the run completed one full epoch over 7.7k prompts (323 optimizer steps total). Reward function latency averaged 23.5s per step (dominated by lean2nl inference).

Table 1: **Mean cycle consistency** (↑) on FLC held-out (700 problems) and PutnamBench (PB, 660 problems). **Cross-entropy evaluation loss** (↓) on FLC validation split (3,932 examples).

Model	FLC $\bar{\mathcal{C}}$	PB $\bar{\mathcal{C}}$	CE loss
Base 2B (few-shot)	0.093	0.034	-
SFT No-Curriculum 2B	0.519	0.432	0.640
SFT Curriculum 2B	0.513	0.422	0.680
RL (GRPO) 2B	**0.669**	**0.561**	0.691
Base 9B (few-shot)	0.029	−0.004	-
SFT Curriculum 9B	0.585	0.548	-

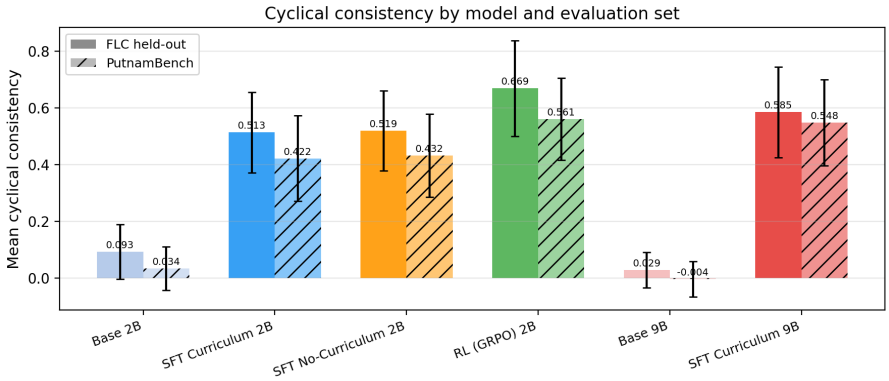


Figure 2: Mean cycle consistency by model and evaluation set. Error bars show ±1 std.

Models are available at `https://huggingface.co/arc-cola` on HuggingFace Hub.

5.4 Results

Results are summarised in Table 1 and Figure 2. RL training substantially improves cycle consistency over both SFT variants on both evaluation sets. RL achieves mean $\bar{\mathcal{C}} = 0.669$, an improvement over SFT curriculum of 0.156 on FLC held-out ($p < 10^{-65}$, Mann-Whitney U) and of 0.139 on PutnamBench ($p < 10^{-56}$, Mann-Whitney U).

The two SFT variants are statistically indistinguishable on both datasets (FLC: $p = 0.46$ and PB: $p = 0.32$, Mann-Whitney U), confirming that curriculum ordering provides no benefit at this model scale or data size.

Notably, SFT curriculum 9B (0.585 FLC, 0.548 PB) outperforms 2B SFT curriculum as expected, but still trails 2B RL (0.669 FLC). On PutnamBench the gap narrows to just 0.013, suggesting that scale partially substitutes for the RL signal on harder problems from a different distribution.

6 Analysis

Score vs. difficulty. Figure 3 shows that cycle consistency is largely flat across difficulty levels 1 to 7 for the SFT models, while the RL model maintains a consistent advantage throughout. Both base models score near zero, confirming that Qwen3.5 base models do not formalize without fine-tuning even with few-shot prompting. The small peak at difficulty 5 across models is likely a dataset artefact (difficulty-5 problems may have more formulaic Lean4 patterns).

RL improvement by difficulty. Figure 4 shows that the RL improvement over SFT curriculum decreases monotonically with difficulty, from an increase of 0.208 at difficulty 1 to 0.126 at difficulty 7. This is consistent with the reward signal being noisier at higher difficulties: harder problems produce worse Lean4 output, which the frozen lean2nl back-translator struggles to interpret, reducing how informative the cosine similarity reward can be.

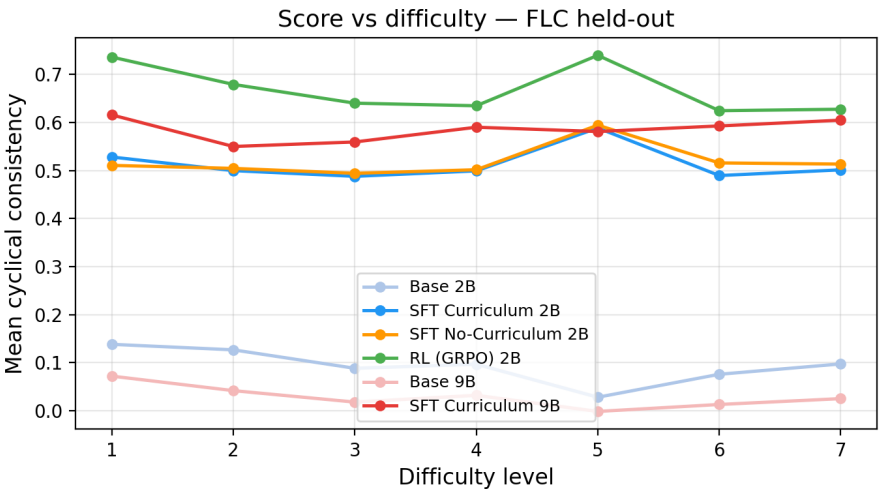


Figure 3: Score vs. difficulty (FLC held-out, 100 problems per difficulty level).

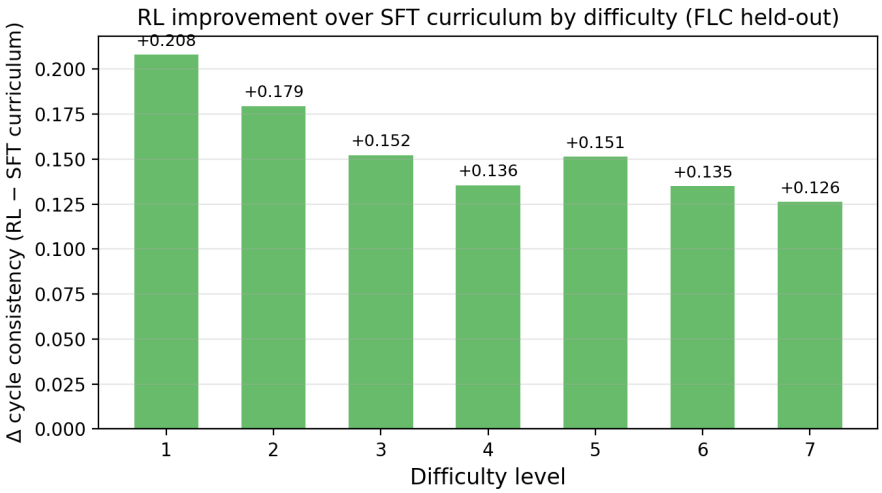


Figure 4: RL improvement over SFT curriculum by difficulty level (FLC held-out). The gain decreases monotonically from 0.208 at difficulty 1 to 0.126 at difficulty 7.

Performance by mathematical domain. Figures 5 and 6 show cycle consistency broken down by mathematical domain. On both datasets a consistent pattern emerges: discrete and algebraic domains (number theory, abstract algebra, algebra, combinatorics) outperform continuous ones (calculus, analysis, probability). On FLC, number theory achieves the highest RL score (0.722) while calculus is the weakest (0.516); on PutnamBench, abstract algebra leads (0.641) while analysis and probability trail (0.493 and 0.508 respectively). Geometry sits mid-table on both datasets (FLC: 0.639, PB: 0.579), contrary to our initial expectation.

I attribute this pattern to the nature of the statements rather than Lean4 library coverage: algebraic and number-theoretic problems use compact, self-contained notation that maps cleanly to Lean4 type signatures and back-translates unambiguously. Continuous mathematics requires ε-δ definitions, limits, and measure-theoretic concepts that produce verbose and structurally complex Lean4 output, making back-translation noisier and the cycle consistency reward less informative. Notably, RL outperforms SFT curriculum in every domain on both datasets, suggesting that the improvement is general rather than domain-specific. Note that the Discrete Mathematics category on FLC ($n = 13$) and Probability on PutnamBench ($n = 14$) are too small for reliable conclusions.

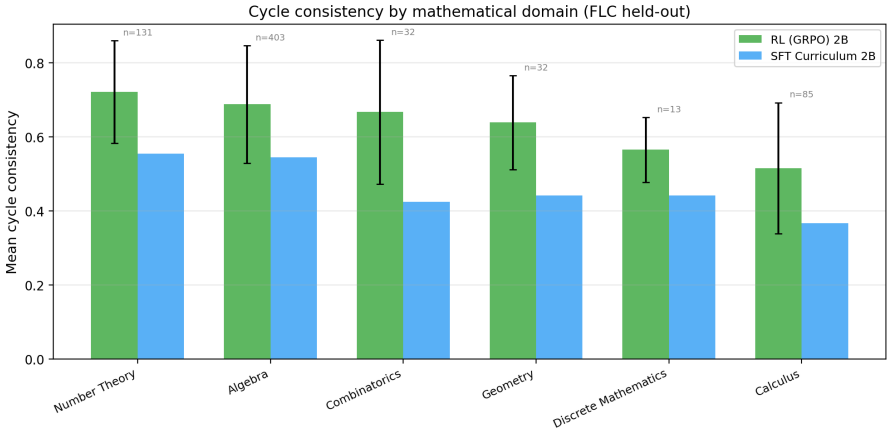


Figure 5: Cycle consistency by mathematical domain on FLC held-out. Error bars show ± 1 std. Domains with $n < 10$ excluded.

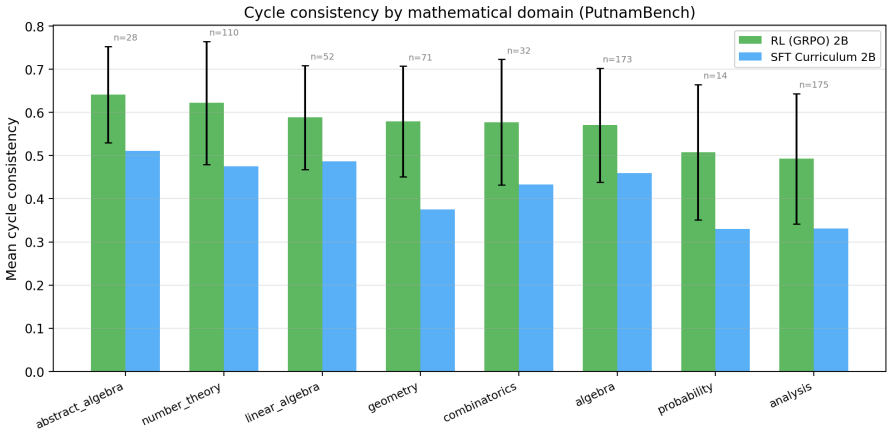
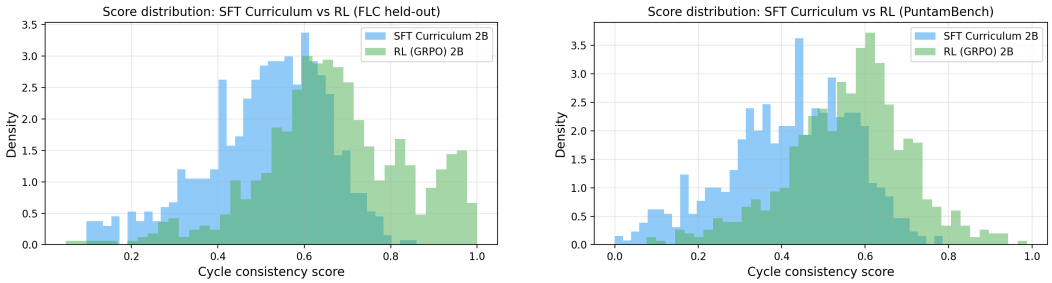


Figure 6: Cycle consistency by mathematical domain on PutnamBench. Error bars show ± 1 std. Domains with $n < 10$ excluded.

Score distribution shift. Figures 7a and 7b compare score distributions for SFT curriculum and RL. On FLC, RL does not simply shift the distribution rightward - it reshapes it, compressing the left tail and creating substantial mass above 0.8 that SFT barely reaches. On PutnamBench, by contrast, RL produces a more uniform rightward shift with the same distributional shape, moving the mean from 0.422 to 0.561 without generating a high-score tail. This contrast suggests that on FLC, RL may be exploiting structural regularities in the dataset. One possible explanation is that many FLC problems share similar Lean4 patterns differing only in variable names or constants, allowing pattern matching to yield high cycle consistency without genuine semantic understanding. Another, observed directly in the data, is that the back-translator occasionally echoes the original statement verbatim when the generated Lean4 closely resembles natural language, producing a perfect reward regardless of formalization quality. Note that these explanations remain speculative without deeper analysis of the FLC dataset. PutnamBench, with its greater problem diversity and out-of-distribution character, is therefore a more reliable benchmark for evaluating the formalization quality (as proxied by the cycle consistency), and the improvement of 0.139 should be considered the more conservative and trustworthy estimate.

GRPO convergence. Figure 8 shows the training reward increasing by 0.06 from 0.82 to 0.88 over the full run. The rate of improvement is highest in the first 100 optimizer steps, after which the curve flattens with a slow upward trend. This rapid early convergence suggests the reward signal is most



(a) FLC held-out.

(b) PutnamBench.

Figure 7: Score distributions: SFT Curriculum 2B vs RL (GRPO) 2B. On FLC, RL reshapes the distribution creating a high-score tail above 0.8. On PutnamBench, RL produces a more uniform rightward shift, suggesting less exploitation of dataset structure.

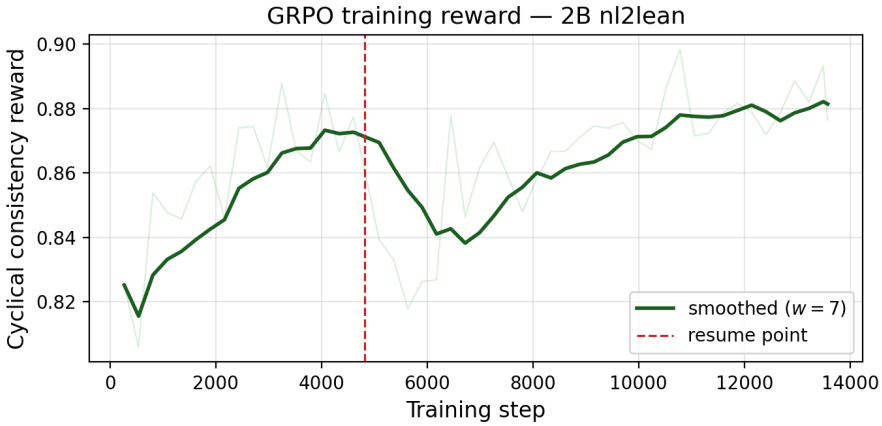


Figure 8: GRPO training reward over the full RL run. Dashed red line marks the checkpoint resume point.

informative when the model has the most headroom to improve from the SFT initialisation. The dip and recovery at the resume point are consistent with optimizer state reconstruction.

Curriculum learning. Figure 9 shows the training token accuracy for curriculum vs. no-curriculum SFT. The curriculum model exhibits periodic resets at difficulty transitions (vertical dotted lines), after which accuracy quickly recovers. Both models converge to similar final token accuracy (between 0.87 and 0.88), and their cycle consistency scores are statistically indistinguishable. This is consistent with findings that curriculum ordering matters less when the model has sufficient capacity and data diversity is achieved via stratification.

Cross-entropy analysis. Figure 10 shows that the three 2B models have nearly identical per-difficulty cross-entropy loss profiles on the FLC val split, with the RL model only marginally higher overall (0.691 vs. 0.680 for SFT curriculum, 0.640 for no-curriculum). The no-curriculum model achieves lowest CE loss, likely because it sees all difficulty levels mixed together which gives more balanced token-level supervision. Interestingly, CE loss decreases from difficulty 1 to 5 then slightly recovers - higher-difficulty problems tend to use more structured Mathlib patterns that are more predictable token-by-token despite being mathematically harder. The marginal RL CE increase (0.011 over SFT curriculum) confirms that GRPO does not significantly degrade the model's formalization ability while improving cycle consistency by 0.156.

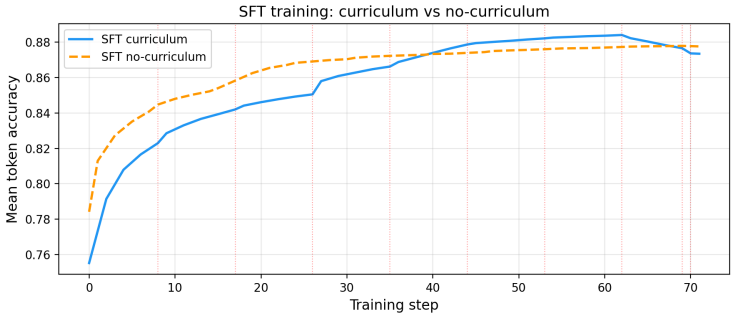


Figure 9: SFT training token accuracy: curriculum (solid) vs. no-curriculum (dashed). Dotted red vertical lines mark difficulty transitions in the curriculum run. No-curriculum x-axis is scaled to match curriculum total steps for comparison.

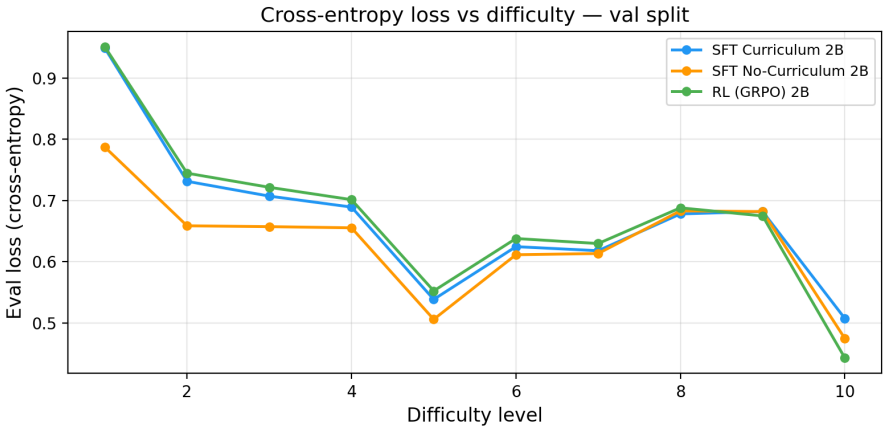


Figure 10: Cross-entropy loss vs. difficulty on FLC val split. All three 2B models track closely.

7 Conclusion

I have shown that GRPO with a cycle consistency reward substantially improves NL $\rightarrow$ Lean4 $\rightarrow$ NL$'$ fidelity for Lean4 autoformalization, performing better than both curriculum and shuffled SFT by a large margin while causing a minimal drop in cross-entropy. Curriculum learning provides no benefit over shuffled SFT at this scale, suggesting that difficulty ordering does not compensate for the limited model capacity of a 2B model on this task. Scale (9B SFT) and RL training (2B GRPO) each improve over 2B SFT, with RL offering a larger boost on the in-distribution evaluation set and scale nearly closing the gap on the harder PutnamBench.

Limitations. Cycle consistency is a proxy metric with several failure modes. First, a model could score high by producing syntactically consistent but mathematically vacuous Lean4; in the statement formalization setting this is compounded by the fact that all generated theorems use `sorry` as a proof placeholder, so even compilation success would indicate only syntactic validity rather than mathematical correctness.Second, the back-translator introduces a consistent bias: using the same frozen lean2nl model for both the RL reward and the evaluation metric means any systematic errors in back-translation affect all models equally, potentially overstating absolute performance. Additionally, direct inspection reveals degenerate cases where the back-translator echoes the original statement verbatim, producing a perfect reward regardless of Lean4 quality - a clear instance of reward hacking enabled by the metric's blind spot.

Finally, I cannot ensure that the base model never saw the PutnamBench dataset, however, the near-zero base model scores on PutnamBench it suggest minimal exposure to Lean4 theorem declarations during pretraining, making data leakage an unlikely problem for our evaluation.

Future work. The most impactful next step is incorporating Lean4 compiler feedback as a hard reward alongside the soft cycle consistency signal. Further directions include scaling RL to 9B, ablating RL initialization from the no-curriculum checkpoint, and replacing the sentence-embedding metric with a stronger semantic equivalence check. Analysing FLC for structural redundancies and evaluating on more diverse held-out sets would help clarify how much of the FLC improvement reflects genuine generalization.

References

[1] Jun-Yan Zhu, Taesung Park, Phillip Isola, and Alexei A Efros. Unpaired image-to-image translation using cycle-consistent adversarial networks. In *ICCV*, 2017.

[2] Zhongyuan Peng, Yifan Yao, Kaijing Ma, Shuyue Guo, Yizhe Li, Yichi Zhang, Chenchen Zhang, Yifan Zhang, Zhouliang Yu, Luming Li, Minghao Liu, Yihang Xia, Jiawei Shen, Yuchen Wu, Yixin Cao, Zhaoxiang Zhang, Wenhao Huang, Jiaheng Liu, and Ge Zhang. CriticLean: Critic-guided reinforcement learning for mathematical formalization, 2025.

[3] George Tsoukalas, Jasper Lee, John Jennings, Jimmy Xin, Michelle Ding, Michael Jennings, Amitayush Thakur, and Swarat Chaudhuri. PutnamBench: Evaluating neural theorem-provers on the Putnam mathematical competition, 2024.

[4] Rico Sennrich, Barry Haddow, and Alexandra Birch. Improving neural machine translation models with monolingual data. In *ACL*, 2016.

[5] Sören Dreano, Derek Molloy, and Noel Murphy. CycleGN: A cycle consistent approach for neural machine translation. In Barry Haddow, Tom Kocmi, Philipp Koehn, and Christof Monz, editors, *Proceedings of the Ninth Conference on Machine Translation*, pages 165–175, Miami, Florida, USA, November 2024. Association for Computational Linguistics.

[6] Jianqiao Wangni. Language models and cycle consistency for self-reflective machine translation, 2024.

[7] Zhanming Shen, Hao Chen, Yulei Tang, Shaolin Zhu, Wentao Ye, Xiaomeng Hu, Haobo Wang, Gang Chen, and Junbo Zhao. CYCLE-INSTRUCT: Fully seed-free instruction tuning via dual self-training and cycle consistency. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2025.

[8] Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, Y. K. Li, Y. Wu, and Daya Guo. DeepSeekMath: Pushing the limits of mathematical reasoning in open language models, 2024.

[9] Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. LoRA: Low-rank adaptation of large language models. In *ICLR*, 2022.

[10] Qwen Team. Qwen3.5: Towards native multimodal agents, February 2026.

Appendix: Qualitative Examples

Here are some examples I hand-picked from the FLC held-out set and PutnamBench to give a feel for what the models actually produce. I split them into three parts: FLC examples, PutnamBench examples, and 9B outputs at the end. The 2B sections only show SFT Curriculum and RL - the base models and no-curriculum variant produce nearly identical outputs to SFT Curriculum and add no narrative value. Each example has the original statement, the ground truth Lean4, and for each model the generated Lean4 and the back-translated NL' with its cycle consistency score.

PutnamBench Examples

Some Putnam problems I found illustrative. The main pattern I kept seeing is that RL gives theorems and hypotheses meaningful names, which helps the back-translator recover the original statement - compare SFT and RL outputs in PB-1, PB-6 and PB-7. PB-10 is a trickier case where both models do fine but the metric still underrewards them.

Example PB-1 [PutnamBench] ***easy***

Original Statement:

> Let a, b, c, d be positive integers satisfying $a + c \leq 1982$ and $\frac{a}{b} + \frac{c}{d} < 1$. Prove that $1 - \frac{a}{b} - \frac{c}{d} > \frac{1}{1983^3}$.

Ground Truth Lean4:

```
import Mathlib

/--
Let $a, b, c, d$ be positive integers satisfying $a + c \leq 1982$ and $\frac{a}{
  {b} + \frac{c}{d} < 1$. Prove that $1 - \frac{a}{b} - \frac{c}{d} > \frac{1}{1
  983^3}$.
-/
theorem putnam_1982_a5
(a b c d : ℤ)
(hpos : a > 0 ∧ b > 0 ∧ c > 0 ∧ d > 0)
(hac : a + c ≤ 1982)
(hfrac : (a : ℝ) / b + (c : ℝ) / d < 1)
  ... [truncated]
```

Model Outputs:

SFT Curriculum 2B (score: 0.590)
Generated Lean4:

```
import Mathlib


theorem qwen (a b c d : ℕ) (_ : 0 < a) (h₀ : 0 < b) (h₁ : 0 < c) (h₂ : 0 < d)
    (h₃ : a + c ≤ 1982) (h₄ : a / b + c / d < 1) :
    1 - a / b - c / d > 1 / 1983 ^ 3 := by sorry
```

Back-translated NL':

```
import Mathlib


theorem qwen (a b c d : ℕ) (ha : 0 < a) (hb : 0 < b) (hc : 0 < c) (hd : 0 < d)
    (h₀ : a + c ≤ 1982) (h₁ : a / b + c / d < 1) :
    1 - a / b - c / d > 1 / 1983 ^ 3 := by sorry
```

RL (GRPO) 2B (score: 0.988)
Generated Lean4:

```
import Mathlib


theorem positive_integers_a_b_c_d (a b c d : ℕ) (ha : 0 < a) (hb : 0 < b) (hc :
  0 < c) (hd : 0 < d)
    (hac : a + c ≤ 1982) (hfrac : a / b + c / d < 1) :
    1 - a / b - c / d > 1 / 1983 ^ 3 := by sorry
```

Back-translated NL':

```
Let $a, b, c, d$ be positive integers such that $a + c \leq 1982$ and $\frac{a}{
  b} + \frac{c}{d} < 1$. Prove that:

\[
1 - \frac{a}{b} - \frac{c}{d} > \frac{1}{1983^3}.
\]
```

Example PB-2 [PutnamBench] ***easy***

Original Statement:

> Find the smallest positive integer n such that for every integer m with $0 < m < 1993$, there exists an integer k for which $\frac{m}{1993} < \frac{k}{n} < \frac{m+1}{1994}$.

Ground Truth Lean4:

```
import Mathlib

abbrev putnam_1993_b1_solution : ℕ := sorry
-- 3987
/--
Find the smallest positive integer $n$ such that for every integer $m$ with $0<m
  <1993$, there exists an integer $k$ for which $\frac{m}{1993}<\frac{k}{n}<\fra
  c{m+1}{1994}$.
-/
theorem putnam_1993_b1 :
    IsLeast
    {n : ℕ | 0 < n ∧
  ... [truncated]
```

Model Outputs:

SFT Curriculum 2B (score: 0.586)
Generated Lean4:

```
import Mathlib

theorem solution :
    IsLeast {n : ℕ | 0 < n ∧ ∀ m : ℤ, 0 < m → m < 1993 → ∃ k : ℤ, m / 1993 < (k
  / n : ℝ) ∧ (k / n : ℝ) < (m + 1) / 1994} 1993 := by sorry
```

Back-translated NL':

```
import Mathlib

theorem solution :
    IsLeast {n : ℕ | 0 < n ∧ ∀ m : ℤ, 0 < m → m < 1993 → ∃ k : ℤ, m / 1993 < (k
  / n : ℝ) ∧ (k / n : ℝ) < (m + 1) / 1994} 1993 := by sorry
```

RL (GRPO) 2B (score: 0.937)
Generated Lean4:

```
import Mathlib
```

```
theorem find_smallest_n :
    IsLeast {n : ℕ | 0 < n ∧ ∀ m : ℤ, 0 < m ∧ m < 1993 → ∃ k : ℤ, m / 1993 < k /
   n ∧ k / n < (m + 1) / 1994} 1993 := by sorry
```

Back-translated NL':

```
prove that the smallest natural number \(n\) such that \(0 < n\) and for all int
  egers \(m\) with \(0 < m < 1993\), there exists an integer \(k\) satisfying \(
  \frac{m}{1993} < \frac{k}{n} < \frac{m + 1}{1994}\) is \(n = 1993\).
```

Example PB-6 [PutnamBench] ***rl-wins***

Original Statement:

> Let (x_n) be a sequence, and let $y_n = x_{n-1} + 2 * x_n$ for $n \geq 2$. Suppose that (y_n) converges, then prove that (x_n) converges.

Ground Truth Lean4:

```
import Mathlib

open Matrix Filter Topology Set Nat

/--
Let $(x_n)$ be a sequence, and let $y_n = x_{n-1} + 2*x_n$ for $n \geq 2$. Suppo
  se that $(y_n)$ converges, then prove that $(x_n)$ converges.
-/
theorem putnam_1969_a6
(x : ℕ → ℝ)
(y : ℕ → ℝ)
(hy1 : ∀ n ≥ 2, y n = x (n-1) + 2 * (x n))
  ... [truncated]
```

Model Outputs:

SFT Curriculum 2B (score: 0.090)
Generated Lean4:

```
import Mathlib

open Topology Filter Real Complex TopologicalSpace Finset
open scoped BigOperators


theorem Tendsto (x : ℕ → ℝ) (y : ℕ → ℝ)
  (h₀ : ∀ n, y n = x (n - 1) + 2 * x n)
  (h₁ : ∃ L, Tendsto y atTop (𝓝 L)) :
  ∃ L, Tendsto x atTop (𝓝 L) :=
sorry
```

Back-translated NL':

```
import Mathlib

open Topology Filter Real Complex TopologicalSpace Finset
```

```
open scoped BigOperators


theorem Tendsto (x y : ℕ → ℝ)
  (h₀ : ∀ n, y n = x (n - 1) + 2 * x n)
  (h₁ : ∃ L, Tendsto y atTop (𝓝 L)) :
  ∃ L, Tendsto x atTop (𝓝 L) :=
sorry
```

RL (GRPO) 2B (score: 0.791)
Generated Lean4:

```
import Mathlib


theorem sequence_convergence (x : ℕ → ℝ) (y : ℕ → ℝ)
    (h_y : ∀ n ≥ 2, y n = x (n - 1) + 2 * x n)
    (h_conv : ∃ L, Tendsto y atTop (𝓝 L)) :
    ∃ L, Tendsto x atTop (𝓝 L) := by sorry
```

Back-translated NL':

```
Let $x: \mathbb{N} \to \mathbb{R}$ and $y: \mathbb{N} \to \mathbb{R}$ be sequenc
  es such that for all $n \geq 2$, $y_n = x_{n-1} + 2x_n$. Suppose that the sequ
  ence $y$ converges to some limit $L$. Prove that the sequence $x$ also converg
  es to some limit $L$.
```

Example PB-7 [PutnamBench] ***rl-wins***

Original Statement:

> Prove that no sequence $\{K_n\}_{n=0}^{\infty}$ of compact (closed and bounded) sets of rational numbers has the property that every compact set of rational numbers is contained by at least one K_n.

Ground Truth Lean4:

```
import Mathlib

open Finset Polynomial Topology Filter Metric

/--
Prove that no sequence $\{K_n\}_{n=0}^{\infty}$ of compact (closed and bounded)
  sets of rational numbers has the property that every compact set of rational n
  umbers is contained by at least one $K_n$.
-/
theorem putnam_1968_b6
: ¬∃ K : ℕ → Set ℚ, (∀ n : ℕ, IsCompact (K n)) ∧ (∀ S : Set ℚ, IsCompact S → ∃ n
   : ℕ, S ⊆ K n) :=
  ... [truncated]
```

Model Outputs:

SFT Curriculum 2B (score: 0.205)
Generated Lean4:

```
import Mathlib

open Topology Filter Real Complex TopologicalSpace Finset
open scoped BigOperators


theorem prove :
  ¬ ∃ (K : ℕ → Set ℚ),
    (∀ n, IsCompact (K n)) ∧
    (∀ S : Set ℚ, IsCompact S → ∃ n, S ⊆ K n) :=
sorry
```

Back-translated NL':

```
import Mathlib

open Topology Filter Real Complex TopologicalSpace Finset
open scoped BigOperators


theorem prove :
  ¬ ∃ (K : ℕ → Set ℚ),
    (∀ n, IsCompact (K n)) ∧
    (∀ S : Set ℚ, IsCompact S → ∃ n, S ⊆ K n) := sorry
```

RL (GRPO) 2B (score: 0.832)
Generated Lean4:

```
import Mathlib


theorem no_sequence_of_compact_sets_of_rational_numbers :
    ¬∃ K : ℕ → Set ℚ,
      (∀ n, IsCompact (K n)) ∧
      (∀ S : Set ℚ, IsCompact S → ∃ n, S ⊆ K n) := by sorry
```

Back-translated NL':

```
prove that there does not exist a sequence of compact sets of rational numbers s
  uch that every compact set of rational numbers is contained in one of the sets
   in the sequence. Formally, prove that there does not exist a function \(K : \
  mathbb{N} \to \text{Set } \mathbb{Q}\) such that for all \(n\), \(K(n)\) is co
  mpact, and for every compact set \(S \subseteq \mathbb{Q}\), there exists an \
  (n\) such that \(S \subseteq K(n)\).
```

Example PB-10 [PutnamBench] ***scale***

Original Statement:

> Suppose that $a_0 = 1$ and that $a_{n+1} = a_n + e^{-a_n}$ for $n = 0, 1, 2, \ldots$. Does $a_n - \log n$ have a finite limit as $n \to \infty$? (Here $\log n = \log_e n = \ln n$.)

Ground Truth Lean4:

```
import Mathlib

open Matrix Function Real Topology Filter

noncomputable abbrev putnam_2012_b4_solution : Prop := sorry
-- True
/--
Suppose that $a_0 = 1$ and that $a_{n+1} = a_n + e^{-a_n}$ for $n=0,1,2,\dots$.
  Does $a_n - \log n$
have a finite limit as $n \to \infty$? (Here $\log n = \log_e n = \ln n$.)
-/
theorem putnam_2012_b4
  ... [truncated]
```

Model Outputs:

SFT Curriculum 2B (score: 0.130)
Generated Lean4:

```
import Mathlib

open Real Filter Function
open scoped Topology


theorem Tendsto (a : ℕ → ℝ)
  (h₀ : a 0 = 1)
  (h₁ : ∀ n, a (n + 1) = a n + exp (-a n)) :
  ∃ L, Tendsto (λ n => a n - log n) atTop (𝓝 L) :=
sorry
```

Back-translated NL':

```
import Mathlib

open Topology Filter Real Complex TopologicalSpace Finset
open scoped BigOperators


theorem Tendsto (a : ℕ → ℝ)
  (h₀ : a 0 = 1)
  (h₁ : ∀ n, a (n + 1) = a n + exp (-a n)) :
  ∃ L, Tendsto (λ n => a n - log n) atTop (𝓝 L) :=
sorry
```

RL (GRPO) 2B (score: 0.220)
Generated Lean4:

```
import Mathlib

open Topology Filter Real Complex TopologicalSpace Finset
open scoped BigOperators


theorem a_n_log_n (a : ℕ → ℝ)
  (h_a0 : a 0 = 1)
  (h_an : ∀ n, a (n + 1) = a n + Real.exp (-a n)) :
  ∃ L, Tendsto (λ n => a n - Real.log n) atTop (𝓝 L) :=
```

```
sorry
```

Back-translated NL':

```
import Mathlib

open Topology Filter Real Complex TopologicalSpace Finset
open scoped BigOperators


theorem a_n_log_n (a : ℕ → ℝ)
  (h_a0 : a 0 = 1)
  (h_an : ∀ n, a (n + 1) = a n + Real.exp (-a n)) :
  ∃ L, Tendsto (λ n => a n - Real.log n) atTop (𝓝 L) :=
sorry
```

FineLeanCorpus (FLC) Held-out Examples

A few examples from my FLC held-out set. I picked these to show the interesting cases - FLC-1 is the reward hacking example where the back-translator just echoes the input, FLC-5 and FLC-6 are cases where RL produces noticeably better Lean4 than SFT.

Example FLC-1 [FLC] ***easy***

Original Statement:

> Prove that for every nonnegative integer k, there exist positive integers a, b, and c such that $a < b < c$ and $k = a^2 + b^2 - c^2$.

Ground Truth Lean4:

```
import Mathlib


theorem prove (k : ℕ) :
    ∃ a b c : ℤ, 0 < a ∧ 0 < b ∧ 0 < c ∧ a < b ∧ b < c ∧ k = a^2 + b^2 - c^2 :=
  by sorry
```

Model Outputs:

SFT Curriculum 2B (score: 0.718)
Generated Lean4:

```
import Mathlib


theorem prove (k : ℕ) :
    ∃ a b c : ℕ, a > 0 ∧ b > 0 ∧ c > 0 ∧ a < b ∧ b < c ∧ k = a ^ 2 + b ^ 2 - c ^
   2 := by sorry
```

Back-translated NL':

```
import Mathlib
```

```
theorem prove (k : ℕ) :
    ∃ a b c : ℕ, a > 0 ∧ b > 0 ∧ c > 0 ∧ a < b ∧ b < c ∧ k = a ^ 2 + b ^ 2 - c ^
   2 := by sorry
```

RL (GRPO) 2B (score: 1.000)
Generated Lean4:

```
import Mathlib


theorem prove_for_every_nonnegative_integer_k (k : ℕ) :
    ∃ a b c : ℕ, a > 0 ∧ b > 0 ∧ c > 0 ∧ a < b ∧ b < c ∧ k = a ^ 2 + b ^ 2 - c ^
   2 := by sorry
```

Back-translated NL':

```
Prove that for every nonnegative integer $k$, there exist positive integers $a$,
   $b$, and $c$ such that $a < b < c$ and $k = a^2 + b^2 - c^2$.
```

Example FLC-5 [FLC] ***rl-wins***

Original Statement:

Prove that $\sum_{i=1}^{n} \binom{i+1}{2} = \binom{n+2}{3}$.

Ground Truth Lean4:

```
import Mathlib

open Finset


theorem Nat.choose_sum (n : ℕ) :
    ∑ i ∈ Icc 1 n, Nat.choose (i + 1) 2 = Nat.choose (n + 2) 3 := by sorry
```

Model Outputs:

SFT Curriculum 2B (score: 0.311)
Generated Lean4:

```
import Mathlib


theorem Nat.choose_sum (n : ℕ) :
    ∑ i in Finset.Icc 1 n, Nat.choose (i + 1) 2 = Nat.choose (n + 2) 3 := by sor
  ry
```

Back-translated NL':

```
import Mathlib


theorem Nat.choose_sum (n : ℕ) :
    ∑ i in Finset.Icc 1 n, Nat.choose (i + 1) 2 = Nat.choose (n + 2) 3 := by sor
  ry
```

RL (GRPO) 2B (score: 0.923)
Generated Lean4:

```
import Mathlib


theorem sum_of_binomial_coefficients (n : ℕ) :
    ∑ i in Finset.Icc 1 n, Nat.choose (i + 1) 2 = Nat.choose (n + 2) 3 := by sor
  ry
```

Back-translated NL':

```
Prove that for any natural number \(n\), the sum of the binomial coefficients \(
  \sum_{i=1}^{n} \binom{i+1}{2}\) is equal to \(\binom{n+2}{3}\).
```

Example FLC-6 [FLC] ***rl-wins***

Original Statement:

> Prove that if a right triangle has integer side lengths a, b, and c such that $a^2 + b^2 = c^2$, then the area of the triangle, given by $\frac{1}{2}ab$, cannot be a perfect square.

Ground Truth Lean4:

```
import Mathlib

theorem area_not_square (a b c : ℤ) (h₀ : a > 0) (h₁ : b > 0) (h₂ : c > 0)
    (h₃ : a^2 + b^2 = c^2) :
    ¬ ∃ n : ℤ, n^2 = (1/2) * a * b := by sorry
```

Model Outputs:

SFT Curriculum 2B (score: 0.419)
Generated Lean4:

```
import Mathlib


theorem prove (a b c : ℤ) (h₀ : a > 0) (h₁ : b > 0) (h₂ : c > 0)
    (h₃ : a^2 + b^2 = c^2) :
    ¬ ∃ m, m^2 = (1 / 2) * a * b := by sorry
```

Back-translated NL':

```
import Mathlib


theorem prove (a b c : ℤ) (h₀ : a > 0) (h₁ : b > 0) (h₂ : c > 0)
    (h₃ : a^2 + b^2 = c^2) :
    ¬ ∃ m, m^2 = (1 / 2) * a * b := by sorry
```

RL (GRPO) 2B (score: 0.952)
Generated Lean4:

```
import Mathlib


theorem prove_area_not_perfect_square (a b c : ℤ) (h_right_triangle : a^2 + b^2
  = c^2)
    (h_right_triangle' : a > 0 ∧ b > 0 ∧ c > 0) :
    ¬ IsSquare (1/2 * a * b) := by sorry
```

Back-translated NL':

```
Prove that for any right triangle with integer side lengths $a$, $b$, and $c$ (w
  here $a^2 + b^2 = c^2$ and $a > 0$, $b > 0$, $c > 0$), the area $\frac{1}{2}ab
  $ is not a perfect square.
```

Example FLC-9 [FLC] ***scale***

Original Statement:

> 1. Prove that if $f : \mathbb{R} \to \mathbb{R}$ is a function such that $f \in C^2$, $f(2) = -1$, $f'(2) = 4$, and $\int_2^3 (3-x)f''(x)dx = 7$, then $f(3) = 10$.
> 2. Since the assistant did not provide a solution for the second problem, we keep the original problem as it is not possible to form a proof statement without a final conclusion. So, the problem remains: Evaluate the integral $\int_0^\infty \frac{\arctan(ax)}{x(1+x^2)}dx$ for $a > 0$.
> 3. Since the assistant did not provide a solution for the third problem, we keep the original problem as it is not possible to form a proof statement without a final conclusion. So, the problem remains: Consider the function $F(x) = \int_0^x (4t^2 - 4t - 1)e^{-t^2+t}dt$. Calculate the value of $\int_0^1 F(x)dx$.

Ground Truth Lean4:

```
import Mathlib


theorem prove_f (f : ℝ → ℝ) (hf : ContDiff ℝ 2 f)
    (h2 : f 2 = -1) (h'2 : deriv f 2 = 4)
    (h'' : ∫ x in (2)..3, (3 - x) * iteratedDeriv 2 f x = 7) :
    f 3 = 10 := by sorry
```

Model Outputs:

SFT Curriculum 2B (score: 0.241)
Generated Lean4:

```
import Mathlib

open Real


theorem prove {f : ℝ → ℝ} (hf : ContDiffOn ℝ 2 f (Set.Icc 2 3))
    (h2 : f 2 = -1) (h'2 : deriv f 2 = 4) (h3 : ∫ x in (2)..3, (3 - x) * iterate
  dDeriv 2 f x = 7) :
    f 3 = 10 := by sorry
```

Back-translated NL':

```
import Mathlib

open Real


theorem prove {f : ℝ → ℝ} (hf : ContDiffOn ℝ 2 f (Set.Icc 2 3))
    (h2 : f 2 = -1) (h'2 : deriv f 2 = 4) (h3 : ∫ x in (2)..3, (3 - x) * iterate
  dDeriv 2 f x = 7) :
    f 3 = 10 := by sorry
```

RL (GRPO) 2B (score: 0.265)
Generated Lean4:

```
import Mathlib


theorem prove_function_f2 (f : ℝ → ℝ)
    (hf : ContDiff ℝ 2 f)
    (h2 : f 2 = -1)
    (h'2 : deriv f 2 = 4)
    (hint : ∫ x in (2)..3, (3 - x) * iteratedDeriv 2 f x = 7) :
    f 3 = 10 := by sorry
```

Back-translated NL':

```
import Mathlib


theorem prove_function_f2 (f : ℝ → ℝ)
    (hf : ContDiff ℝ 2 f)
    (h2 : f 2 = -1)
    (h'2 : deriv f 2 = 4)
    (hint : ∫ x in (2)..3, (3 - x) * iteratedDeriv 2 f x = 7) :
    f 3 = 10 := by sorry
```

SFT Curriculum 9B Outputs

The 9B model was run with the same prompt as the 2B models, but it kept generating mathematical reasoning in natural language instead of Lean4. My best guess is that its stronger reasoning capabilities caused it to ignore the output format constraints, possibly with thinking mode active. I'm including these outputs anyway since they show an interesting failure mode at larger scale.

PutnamBench Examples **Example PB-1 [PutnamBench]** ***easy***

> Let a, b, c, d be positive integers satisfying $a + c \leq 1982$ and $\frac{a}{b} + \frac{c}{d} < 1$. Prove that $1 - \frac{a}{b} - \frac{c}{d} > \frac{1}{1983^3}$.

SFT Curriculum 9B (score: 0.738)
Generated Lean4:

```
Let the given conditions be:
1. $a, b, c, d \in \mathbb{Z}^+$
2. $a + c \leq 1982$
3. $\frac{a}{b} + \frac{c}{d} < 1$
```

```
We want to prove that $1 - \frac{a}{b} - \frac{c}{d} > \frac{1}{1983^3}$.

Let's rewrite the inequality we want to prove:
$1 - \frac{a}{b} - \frac{c}{d} > \frac{1}{1983^3}$
$\frac{bd - ad - bc}{bd} > \frac{1}{1983^3}$
$1983^3 (bd - ad - bc) > bd$

  ... [truncated]
```

Back-translated NL':

```
$.
Since $\frac{c}{d} < 1$, we have $d > c$.
Also $a + c \leq 1982$.
So $b > a$ and $d > c$ and $a + c \leq 1982$.
We want to show that $\frac{bd - ad - bc}{bd} > \frac{1}{1983^3}$.
This is equivalent to $1983^3 (bd - ad - bc) > bd$.
Let's consider the maximum possible value of $bd$.
Since $b > a$ and $d > c$ and $a + c \leq 1982$, the maximum value of $b$ is $19
  82$ and the maximum value of $d$ is $1982$.
So the maximum value of $bd$ is $1982^2$.
We want to show that $1983^3 (bd - ad - bc) > bd$.
Since $bd \leq 1982^2$, we have $1983^3 (bd - ad - bc)
```

Example PB-2 [PutnamBench] ***easy***

> Find the smallest positive integer n such that for every integer m with $0 < m < 1993$, there exists an integer k for which $\frac{m}{1993} < \frac{k}{n} < \frac{m+1}{1994}$.

SFT Curriculum 9B (score: 0.758)
Generated Lean4:

```
The user wants to find the smallest positive integer $n$ that satisfies a specif
  ic condition involving inequalities.

Let the condition be:
For every integer $m$ with $0 < m < 1993$, there exists an integer $k$ such that
$$ \frac{m}{1993} < \frac{k}{n} < \frac{m+1}{1994} $$

We need to find the minimum $n \in \mathbb{Z}^+$ such that this holds.

Let's analyze the inequality:
$$ \frac{m}{1993} < \frac{k}{n} < \frac{m+1}{1994} $$
This is equivalent to:
  ... [truncated]
```

Back-translated NL':

```
\frac{mn}{1993}, \frac{(m+1)n}{1994})$, the length of the interval must be great
  er than 1.
So, we need:
$$ n \left( \frac{1993 - m}{1993 \cdot 1994} \right) > 1 $$
$$ n > \frac{1993 \cdot 1994}{1993 - m} $$

This must hold for all $m$ with $0 < m < 1993$.
```

```
The maximum value of $\frac{1993 \cdot 1994}{1993 - m}$ occurs when $m$ is as la
  rge as possible, i.e., $m = 1992$.
$$ n > \frac{1993 \cdot 1994}{1993 - 1992} = 1993 \cdot 1994 = 3974042 $$

So, the smallest positive integer $n$ that satisfies the condition is $n = 39740
  ... [truncated]
```

Example PB-6 [PutnamBench] ***rl-wins***

> Let (x_n) be a sequence, and let $y_n = x_{n-1} + 2 * x_n$ for $n \geq 2$. Suppose that (y_n) converges, then prove that (x_n) converges.

SFT Curriculum 9B (score: 0.598)
Generated Lean4:

```
Here's a thinking process that leads to the Lean4 formalization:

1.  **Analyze the Problem Statement:**
    *   **Given:** A sequence $(x_n)$ (implicitly indexed by natural numbers, us
  ually starting from 1 or 0, but the relation $y_n = x_{n-1} + 2x_n$ for $n \ge
  q 2$ suggests indices $\geq 1$ or $\geq 0$).
    *   **Definition:** A new sequence $(y_n)$ is defined by $y_n = x_{n-1} + 2x
  _n$ for $n \geq 2$.
    *   **Hypothesis:** The sequence $(y_n)$ converges.
    *   **Conclusion:** The sequence $(x_n)$ converges.

2.  **Determine the Indexing Convention:**
  ... [truncated]
```

Back-translated NL':

```
: \mathbb{N} \to \mathbb{R}$ such that for all $n \geq 2$, $y_n = x_{n-1} + 2x_n
  $.
    *   The hypothesis is that the sequence $(y_n)$ converges. In Lean, this is
  `∃ L, Tendsto y atTop (𝓝 L)`.

4.  **Formalize the Conclusion:**
    *   The conclusion is that the sequence $(x_n)$ converges. In Lean, this is
  `∃ L, Tendsto x atTop (𝓝 L)`.

5.  **Refine the Definition of $y$:**
    *   The problem statement says $y_n = x_{n-1} + 2x_n$ for $n \geq 2$.
    *   We can define $y$ on all of $\mathbb{N}$ arbitrarily for $n < 2$, or jus
  ... [truncated]
```

Example PB-7 [PutnamBench] ***rl-wins***

> Prove that no sequence $\{K_n\}_{n=0}^{\infty}$ of compact (closed and bounded) sets of rational numbers has the property that every compact set of rational numbers is contained by at least one K_n.

SFT Curriculum 9B (score: 0.688)
Generated Lean4:

```
Here's a thinking process that leads to the formalization of the statement:
```

```
1.  **Analyze the original statement:**
    *   **Goal:** Prove that no sequence $\{K_n\}_{n=0}^{\infty}$ of compact (cl
  osed and bounded) sets of rational numbers has the property that every compact
   set of rational numbers is contained by at least one $K_n$.
    *   **Key terms:**
        *   "Sequence of sets": A function from $\mathbb{N}$ to the power set of
   $\mathbb{Q}$.
        *   "Compact sets of rational numbers": In the context of $\mathbb{Q}$ (
  with the subspace topology from $\mathbb{R}$), a set is compact if and only if
   it is closed and bounded.
  ... [truncated]
```

Back-translated NL':

```
Lean4:**
    *   We need to state that there does *not* exist a sequence $K$ such that ea
  ch $K_n$ is compact and every compact set is contained in some $K_n$.
    *   The statement "Every compact set of rational numbers is contained by at
  least one $K_n$" translates to: $\forall S \subseteq \mathbb{Q}, \text{IsCompa
  ct } S \to \exists n, S \subseteq K_n$.
    *   The statement "No sequence $\{K_n\}$ has the property..." translates to:
   $\neg \exists K : \mathbb{N} \to \text{Set } \mathbb{Q}, (\forall n, \text{Is
  Compact } K_n) \land (\forall S, \text{IsCompact } S \to \exists n, S \subsete
  q K_n)$.

4.  **Final Polish:**
  ... [truncated]
```

Example PB-10 [PutnamBench] ***scale***

> Suppose that $a_0 = 1$ and that $a_{n+1} = a_n + e^{-a_n}$ for $n = 0, 1, 2, \ldots$. Does $a_n - \log n$ have a finite limit as $n \to \infty$? (Here $\log n = \log_e n = \ln n$.)

SFT Curriculum 9B (score: 0.823)
Generated Lean4:

```
The user wants to determine if the sequence $a_n - \log n$ has a finite limit as
   $n \to \infty$, given the recurrence relation $a_0 = 1$ and $a_{n+1} = a_n +
  e^{-a_n}$.

Let's analyze the recurrence relation.
$a_{n+1} - a_n = e^{-a_n}$.
Since $a_0 = 1 > 0$, and $e^{-a_n} > 0$, the sequence $a_n$ is strictly increasi
  ng.
Also, $a_{n+1} > a_n$, so $a_n \to \infty$ as $n \to \infty$.
We want to check the limit of $a_n - \log n$.

Let's approximate the recurrence.
  ... [truncated]
```

Back-translated NL':

```
So $a_n - \log n \to 0$ as $n \to \infty$.

To make this rigorous, we can use the Stolz-Cesàro theorem or other asymptotic a
```

```
  nalysis techniques.
But the key idea is that $a_n$ grows like $\log n$.

So the limit of $a_n - \log n$ is finite (specifically, it is 0).

The final conclusion is that the sequence $a_n - \log n$ has a finite limit as $
  n \to \infty$.

So the answer is \boxed{0}.
  ... [truncated]
```

FLC Examples **Example FLC-1 [FLC]** ***easy***

> Prove that for every nonnegative integer k, there exist positive integers a, b, and c such that $a < b < c$ and $k = a^2 + b^2 - c^2$.

SFT Curriculum 9B (score: 0.896)
Generated Lean4:

```
Let me analyze this problem step by step.

We need to prove that for every nonnegative integer $k$, there exist positive in
  tegers $a$, $b$, and $c$ such that $a < b < c$ and $k = a^2 + b^2 - c^2$.

Let me think about this. We want to find $a, b, c \in \mathbb{Z}^+$ with $a < b
  < c$ such that $k = a^2 + b^2 - c^2$.

Rearranging, we get $c^2 = a^2 + b^2 - k$.

So we need to find positive integers $a, b, c$ with $a < b < c$ such that $c^2 =
   a^2 + b^2 - k$.
  ... [truncated]
```

Back-translated NL':

```
we can ensure the inequality $a < b < c$ holds while satisfying the equation $k
  = a^2 + b^2 - c^2$.

The construction involves selecting appropriate values for $a$, $b$, and $c$ bas
  ed on the given $k$, ensuring all conditions are met.

Ultimately, we can prove that for any nonnegative integer $k$, there exist posit
  ive integers $a$, $b$, and $c$ with $a < b < c$ such that $k = a^2 + b^2 - c^2
  $.
assistant
<think>

  ... [truncated]
```

Example FLC-5 [FLC] ***rl-wins***

> Prove that $\sum_{i=1}^{n} \binom{i+1}{2} = \binom{n+2}{3}$.

SFT Curriculum 9B (score: 0.293)
Generated Lean4:

```
Here's a thinking process that leads to the formal math proof statement:

1.  **Analyze the Request:**
    *   **Input:** A mathematical identity: $\sum_{i=1}^n \binom{i+1}{2} = \bino
  m{n+2}{3}$.
    *   **Task:** Formalize this statement in Lean4.

2.  **Understand the Mathematical Objects:**
    *   The expression involves a summation $\sum_{i=1}^n$.
    *   It involves binomial coefficients $\binom{n}{k}$, often denoted as `Nat.
  choose n k` in Lean.
    *   The variable $n$ is a natural number (implicitly $n \ge 1$ for the sum t
  ... [truncated]
```

Back-translated NL':

```
∈ Finset.Icc 1 n, Nat.choose (i + 1) 2 = Nat.choose (n + 2) 3`

5.  **Refine the Statement:**
    *   The statement looks correct and standard.
    *   It captures the essence of the input identity.

6.  **Final Polish:**
    *   The user asked for a "formal math proof statement".
    *   The output should be the Lean4 code block.

    *Self-Correction/Verification:*
    *   Does `Finset.Icc 1 n` work for $n=0$?
  ... [truncated]
```

Example FLC-6 [FLC] ***rl-wins***

> Prove that if a right triangle has integer side lengths a, b, and c such that $a^2 + b^2 = c^2$, then the area of the triangle, given by $\frac{1}{2}ab$, cannot be a perfect square.

SFT Curriculum 9B (score: 0.750)
Generated Lean4:

```
Let me analyze this problem step by step.

We have a right triangle with integer side lengths $a$, $b$, and $c$ where $a^2
  + b^2 = c^2$.

We need to prove that the area $\frac{1}{2}ab$ cannot be a perfect square.

Let me think about this. If $\frac{1}{2}ab = k^2$ for some integer $k$, then $ab
   = 2k^2$.

Since $a^2 + b^2 = c^2$, we have a Pythagorean triple.

  ... [truncated]
```

Back-translated NL':

```
$s$, then we have a contradiction.

Actually, let me think about this problem from a different angle. Let's use the
  fact that if $a^2 + b^2 = c^2$, then we can write $a = k(m^2 - n^2)$, $b = k(2
  mn)$, $c = k(m^2 + n^2)$ for some integers $k, m, n$.

The area is $\frac{1}{2}ab = k^2mn(m^2 - n^2)$.

For this to be a perfect square, we need $mn(m^2 - n^2)$ to be a perfect square.

Let me think about this more carefully. If $mn(m^2 - n^2) = s^2$ for some intege
  r $s$, then we have a contradiction.
  ... [truncated]
```

Example FLC-9 [FLC] ***scale***

> 1. Prove that if $f : \mathbb{R} \to \mathbb{R}$ is a function such that $f \in C^2$, $f(2) = -1$, $f'(2) = 4$, and $\int_2^3 (3 - x)f''(x)dx = 7$, then $f(3) = 10$.
> 2. Since the assistant did not provide a solution for the second problem, we keep the original problem as it is not possible to form a proof statement without a final conclusion. So, the problem remains: Evaluate the integral $\int_0^\infty \frac{\arctan(ax)}{x(1+x^2)}dx$ for $a > 0$.
> 3. Since the assistant did not provide a solution for the third problem, we keep the original problem as it is not possible to form a proof statement without a final conclusion. So, the problem remains: Consider the function $F(x) = \int_0^x (4t^2 - 4t - 1)e^{-t^2+t}dt$. Calculate the value of $\int_0^1 F(x)dx$.

SFT Curriculum 9B (score: 0.874)
Generated Lean4:

```
The user wants me to formalize three mathematical problems into Lean4.

**Problem 1:**
*   **Given:**
    *   \(f: \mathbb{R} \to \mathbb{R}\) is a function.
    *   \(f \in C^{2}\) (twice continuously differentiable).
    *   \(f(2) = -1\).
    *   \(f'(2) = 4\).
    *   \(\int_{2}^{3}(3 - x)f''(x)dx = 7\).
*   **Prove:** \(f(3) = 10\).

**Problem 2:**
  ... [truncated]
```

Back-translated NL':

```
Let's re-read the user's prompt. The user's prompt is:
    1.  Prove that if \(f: \mathbb{R} \to \mathbb{R}\) is a function such that \
  (f \in C^{2}\), \(f(2)=-1\), \(f'(2)=4\), and \(\int_{2}^{3}(3 - x)f''(x)dx =
  7\), then \(f(3)=10\).
    2.  Evaluate \(\int_{0}^{\infty}\frac{\arctan(ax)}{x(1 + x^{2})}dx\) for \(a
   > 0\).
    3.  Consider the integral \(\int_{0}^{\infty}\frac{\arctan(ax)}{x(1 + x^{2})
  }dx\) for \(a > 0\).
*   The user's prompt is a bit ambiguous. It seems like they want me to formaliz
```

```
  e the problems.
*   For problem 1, it's already a formal proof statement.
*   For problem 2 and 3, they are not proof statements
```