
Agentic Lean Autoformalization (ALA): An LLM collaborative approach to autoformalization in LEAN

Anonymous Author(s)

Affiliation

Address

email

Abstract

1 The arrival of AI systems that can achieve a gold medal at the International Mathe-
2 matical Olympiad (IMO) and the development of proof assistants such as Lean seem
3 to foretell a transformative revolution in mathematical research. However, a bottle-
4 neck is that most undergraduate- and graduate-level theorems are not translated
5 into code for proof assistants, a process known as *autoformalization*. State-of-
6 the-art fine-tuned LLMs in Lean 4 report at most 22.5% accuracy (Pass@128) on
7 graduate-level theorems. To address this gap, we propose and evaluate ALA, an
8 agentic framework where a generalist LLM orchestrating tools works together with
9 another LLM fine-tuned in Lean 4. ALA achieves a 52 % accuracy with less than
10 13 tool-calls on theorems from areas such as complex and real analysis, topology,
11 and algebra. Our code and the related dataset are published on GitHub. ¹

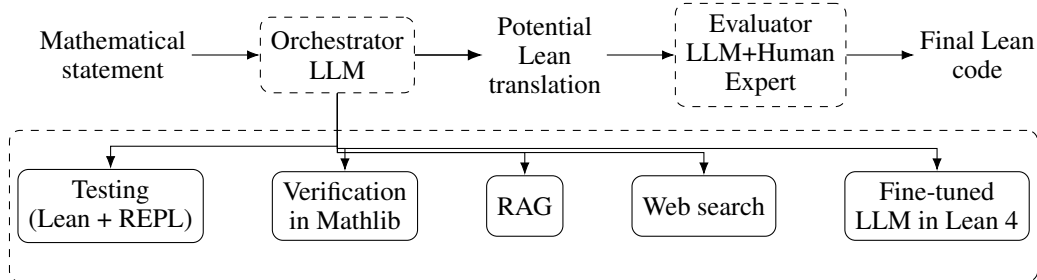
12 1 Introduction

13 Although large language models (LLMs) are increasingly capable of producing complex mathematical
14 arguments [Cas25], their probabilistic outputs conflict with the certainty required by the mathematical
15 community. Proof assistants such as Lean [dMU21] address this conflict by formally certifying
16 the logical correctness of a proposed proof. The transformative nature of combining generative
17 AI with formal verification has recently attracted much attention within mathematical research
18 [BAMa, BAMb]. In particular, there is an increasing number of fine-tuned LLMs trained on Lean 4
19 data and autoformalization [GWJ⁺25] [WUL⁺25] [WZJ⁺24]. However, the use of such tools for the
20 working mathematician is currently limited because many important undergraduate- and graduate-
21 level topics, such as the special linear group, algebras over commutative rings, are not yet available in
22 Lean code [Lea]. We discuss some of the challenges of autoformalization in Section 2 and Appendix
23 A.1.

24 **Contributions:** Our contributions to address this challenge are threefold. (i) We present ALA,
25 Agentic Lean Autoformalization, an agentic framework that combines the abilities of a fine-tuned
26 LLM in Lean 4, the tool capabilities of a generalist LLM, and a combination of human expert and
27 LLM judgment for translating mathematical statements to Lean 4, see Figure 1, (ii) We present a
28 database of 200 graduate and 200 upper undergraduate level theorems covering topology, analysis,
29 algebra, real and complex analysis. (iii) We evaluate ALA and identify strengths, weaknesses, and
30 future areas of work on our agentic approach. ALA translates 64% of the 400 problems in our
31 database with less than 25 tool calls. Results are discussed in Section 6.

¹<https://anonymous.4open.science/r/LeanTranslationAgent-CC41/README.md>

Figure 1: ALA framework: A generalist LLM with access to four tools, a LLM fine-tuned on Lean 4, and a reasoning LLM model that, together with a human expert, evaluates the final accuracy of the translation, see Section 3.



32 2 Preliminaries

33 2.1 Autoformalization

34 The goal of *autoformalization* is to automatically translate mathematics from natural language into
 35 machine-checked formal code. This vision dates back at least to de Bruijn’s *AUTOMATH* [dB70]
 36 and has seen a modern resurgence with LLMs and interactive theorem provers. AI is the tool for
 37 automation, whereas proof assistants—here, Lean 4—are the setting for formalization. Currently,
 38 there is active research on improving LLMs to generate proofs in Lean 4 and on constructing databases
 39 for future AI training; see [WDL⁺25]. We highlight three key challenges.

40 (1) Translating a theorem statement into compiling Lean 4 code—even without a proof—depends
 41 on prior notations, typeclass instances, definitions, and lemmas. For example, we cannot formalize
 42 vector spaces without a previous formalization of a field such as the real numbers.

43 (2) Generating Lean 4 code that compiles does not guarantee, without human evaluation, that the
 44 translation is faithful. Sometimes, the errors are obvious and in other cases they are much subtler, see
 45 the Appendix A.1.

46 (3) Lean 4 is based on an extension of Martin–Löf’s dependent type theory [dMU21], whereas
 47 traditional mathematics is based on an extension of set theory. These foundations are radically
 48 different; for example, in type theory, even proofs are first-class citizens part of the object-language,
 49 but in set theory, a proof is part of the meta-language and not naively the object-language.

50 2.2 Agents

51 In classical AI, an *agent* perceives and acts to achieve goals; modern LLM-based agents extend this
 52 loop by interleaving planning, tool use, observation, and revision [RN95, GCW⁺24]. In software
 53 engineering, multi-agent systems coordinate specialized roles for retrieval, coding, execution, and
 54 debugging [HTL25]. Such agents can also *self-reflect*, storing intermediate attempts and feedback to
 55 guide subsequent decisions [SCB⁺24]. Given these advantages, in formalization an agentic approach
 56 that couples a generalist planner with Lean-specialized models and treats the proof assistant as a
 57 verifier is natural: it can decompose natural-language statements, retrieve examples, autoformalize
 58 necessary lemmas, and iterate. Recent work further augments this pattern [BLKS25, SYA25]. Coding
 59 and reasoning agents still struggle to sustain verifiable control over long action chains with external
 60 tools—planning, executing, and repairing across dozens of steps.

61 3 ALA framework

62 Our Agentic Lean Autoformalization (ALA) framework is centered around a generalist large language
 63 model (LLM) orchestrator that has access to a Lean 4-specialized model and multiple tools to improve
 64 the reliability of autoformalization. The orchestrator has three core abilities:

- 65 (i) **Search for information and context:** The orchestrator can use `lean_retrieval` to
 66 fetch context and examples from a dedicated database that consists of theorems in natural

67 language, their translations to Lean 4, and explanations of the translations. Additionally,
68 the orchestrator can use `search_online` to search the web for Lean 4-related code or
69 documentation.

70 (ii) **Collect feedback:** The orchestrator can use `lean4_repl_runner` to compile Lean
71 code and collect diagnostics via the REPL package [Com24]. It can also use the tool
72 `check_theorem_tool` to construct a temporary Lean file, import `Mathlib`, and use the
73 `#check` command to inspect the type of a definition, expression, or theorem.

74 (iii) **Query an expert:** The orchestrator can use `lean4_translation` to produce a Lean 4
75 declaration from natural language by prompting an LLM that has been fine-tuned in Lean 4.

76 Given a mathematical statement in natural language, the orchestrator interacts with the above resources
77 until it produces Lean 4 code that compiles without errors or the number of tool calls reaches a bound
78 given by the user. It then exports the Lean code. At this point, the candidate translation is sent to
79 a reasoning-model LLM and presented to the user, who is assumed to be knowledgeable about the
80 mathematical aspects of the definition and able to identify mathematically equivalent definitions
81 written in different forms. The Lean code can be approved as a translation, rejected, or sent back
82 to the orchestrator with feedback for future work by combining the LLM evaluation with a human
83 evaluation as well.

84 4 A new database of upper-level theorems

85 There are several well-known databases of theorems produced by the autoformalization community.
86 For example, `FineLeanCorpus` [m-a25, PYM⁺25] contains 509,356 pairs of natural language with
87 Lean 4 code; 1,181 from `Omni-MATH` [GSY⁺24] (undergraduate and olympiad) and 45,853 from
88 `DeepMath-103K`.

89 However, our agent has access to web-search, so to avoid contamination we exclude common datasets
90 with informal mathematics whose statements already appear paired with Lean 4 code. [HLX⁺25]. To
91 minimize collisions, we chose examples from freely accessible repositories written by professors: Jiří
92 Lebl’s *Basic Analysis* and *Guide to Cultivating Complex Analysis* [Leb25a, Leb25b], Ben McKay’s
93 lecture notes on topology [McK25], and Stephen Doty’s *Lecture Notes on Abstract Algebra* [Dot25].
94 For each source, we selected 100 examples by diversifying topic area and length. In total, our database
95 consists of 400 examples, split evenly across undergraduate real, complex analysis, topology, and
96 algebra.

97 5 Experimental setup

98 We evaluate ALA on our corpus of $N = 400$ theorems in algebra, topology, real analysis, and
99 complex analysis, see Section 4. For each natural-language statement, the task is to produce a Lean 4
100 statement that type-checks in `Mathlib` and that it’s a faithful translation of the initial mathematical
101 statement. Next, we describe the particularities of our experiments. We used Lean 4.22.0-rc4 compiler
102 and `mathlib4` as dependency.

103 **Settings to test:** We compare three settings with a budget of 24 tool calls per problem. The baseline
104 setting is the orchestrator LLM with access to all the tools described in Section 3. In our first variation,
105 we modify the baseline setting by removing access to the LLM fine-tuned on Lean. In our second
106 variation, we remove access to all tools besides the LLM fine-tuned on Lean 4 and the ability of the
107 orchestrator to tell if a given Lean code compiles.

108 All methods use the same prompts and inputs. We record the number of calls used until orchestrator
109 produces a Lean 4 statement that compiles; we also record the number of tool calls. We report pass
110 rates, area-wise stratification, and $\text{Pass}@k$ over $k \in \{1, 6, 24\}$. We reset tool states between methods,
111 fix random seeds, and log tool traces for paired analysis.

112 **Model selection:** For the generalist model, we use OpenAI 5.1 mini. For the LLM fine-tuned on
113 Lean 4, we use Herald Translator [GWJ⁺25]. For the final evaluation, we use the OpenAI 5.1 model.

114 **Databases:** For retrieving examples, we use a subset of 500 statements from the Herald database
115 [GWJ⁺25]. For testing, we use our database, see Section 4.

116 **Evaluation metrics:** We use the proportion of theorems that the agent successfully translated with
 117 fewer than $(K + 1)$ tools. We also consider the proportions of potential translations that compile as
 118 Lean code, but they may not be mathematically equivalent to the original statement.

119 6 Discussion of Experimental results

120 We found that an agentic approach is successful for translating mathematical statements to Lean. In
 121 particular, the use of tools had an impact on the success rate, on problems that require more tool calls
 122 to be translated, see Table 1. The full agent configuration translates 64 % of the theorems within 24
 123 tool calls. This shows a significant improvement over the performance of Herald translator, 23%, 16%
 124 (Pass 128), and of Theorem Llama, 4 % and 2.9 % (Pass 128) for problems of a similar mathematical
 125 level.

Table 1: Success rate $SR@K$ for autoformalization $\pm 95\%$ Confidence interval

Number of tools called	Agent with tools and expert LLM	Agent with tools but without expert LLM	Agent with expert LLM but without tools
5	0.2050 ± 0.0422	0.2100 ± 0.0426	0.1950 ± 0.0416
10	0.3950 ± 0.0486	0.4375 ± 0.0489	0.3425 ± 0.0478
15	0.5225 ± 0.0485	0.5650 ± 0.0478	0.4150 ± 0.0489
20	0.6100 ± 0.0465	0.6100 ± 0.04654	0.4750 ± 0.049
24	0.6400 ± 0.0455	0.6725 ± 0.0442	0.5575 ± 0.0417

126 We also fit a Cox proportional hazards model to time-to-event data with a single binary indicator: the
 127 agent has access to all its tool configurations A.3. We found that the data is compatible with anything
 128 from a modest decrease (about 8.5%) to a moderate increase (about 36.4%), see Table 2.

129 6.1 Limitations

- 130 1. We have tried to minimize contamination; parts of our evaluation set may already be
 131 formalized on the web. Thus, the agent could “cheat” by retrieving solutions. Although the
 132 pipeline provides logs, we have not fully analyzed these mistakes.
- 133 2. Similar to Herald Translator, we use a baseline LLM call to judge faithfulness. The choice
 134 of this LLM matters and can yield false positives/negatives (see Appendix A.2). In future
 135 work we will consider specialist judges such as CriticLeanGPT [PYM⁺25].
- 136 3. The agent prompt can be further optimized, the dataset enlarged, and the RAG database
 137 extended to the full Herald set.
- 138 4. We were bottlenecked by compile checks: whenever the ALA calls certain tools, the
 139 loop blocks until they finish. In particular, `run_repl_tool` and `check_theorem_tool`
 140 dominate runtime— a Lean 4 REPL check takes about 30 seconds in our setup, and since
 141 `check_theorem_tool` invokes the REPL, it inherits the same cost.

142 References

- 143 [BAMa] Bull. amer. math. soc. (n.s.). Chief editor articles.
- 144 [BAMb] Bulletin of the american mathematical society.
- 145 [BLKS25] Kaito Baba, Chaoran Liu, Shuhei Kurita, and Akiyoshi Sannai. Prover agent: An agent-
 146 based framework for formal mathematical proofs. *arXiv preprint arXiv:2506.19923*,
 147 2025.
- 148 [Cas25] Davide Castelvechi. Ai models solve maths problems at level of top students. *Nature*,
 149 644:7, 2025.
- 150 [Com24] Leanprover Community. repl: a REPL for Lean 4. [https://
 151 github.com/leanprover-community/repl] (https://github.com/
 152 leanprover-community/repl), 2024.

- 153 [dB70] N. G. de Bruijn. The mathematical language automath, its usage, and some of its
154 extensions. In M. Laudet, D. Lacombe, L. Nolin, and M. Schützenberger, editors,
155 *Proceedings Symposium on Automatic Demonstration (Versailles, France, December*
156 *1968)*, volume 125 of *Lecture Notes in Mathematics*, pages 29–61. Springer, Berlin,
157 1970.
- 158 [dMU21] Leonardo de Moura and Sebastian Ullrich. The lean 4 theorem prover and programming
159 language (system description). In *Automated Deduction – CADE 28*, volume 12699 of
160 *LNCS*, pages 625–635. Springer, 2021.
- 161 [Dot25] Stephen R. Doty. Lecture notes on abstract algebra. [https://github.com/srdoty/](https://github.com/srdoty/AbstractAlgebraBook)
162 [AbstractAlgebraBook](https://github.com/srdoty/AbstractAlgebraBook), 2025. GitHub repository; accessed 2025-09-04.
- 163 [GCW⁺24] Taicheng Guo, Xiuying Chen, Yaqi Wang, Ruidi Chang, Shichao Pei, Nitesh V Chawla,
164 Olaf Wiest, and Xiangliang Zhang. Large language model based multi-agents: A survey
165 of progress and challenges. In *IJCAI*, 2024.
- 166 [GSY⁺24] Bofei Gao, Feifan Song, Zhe Yang, Zefan Cai, Yibo Miao, Qingxiu Dong, Chenghao Ma,
167 Liang Chen, Runxin Xu, Zhengyang Tang, Benyou Wang, Daoguang Zan, Shanghaoran
168 Quan, Ge Zhang, Lei Sha, Yichang Zhang, Xuancheng Ren, Tianyu Liu, and Baobao
169 Chang. Omni-math: A universal olympiad level mathematic benchmark for large
170 language models. *arXiv preprint arXiv:2410.07985*, 2024.
- 171 [GWJ⁺25] Guoxiong Gao, Yutong Wang, Jiedong Jiang, Qi Gao, Zihan Qin, Tianyi Xu, and
172 Bin Dong. Herald: A natural language annotated lean 4 dataset. In *The Thirteenth*
173 *International Conference on Learning Representations*, 2025.
- 174 [HLX⁺25] Zhiwei He, Tian Liang, Jiahao Xu, Qiuzhi Liu, Xingyu Chen, Yue Wang, Linfeng Song,
175 Dian Yu, Zhenwen Liang, Wenxuan Wang, Zhuosheng Zhang, Rui Wang, Zhaopeng
176 Tu, Haitao Mi, and Dong Yu. Deepmath-103k: A large-scale, challenging, decontam-
177 inated, and verifiable mathematical dataset for advancing reasoning. *arXiv preprint*
178 *arXiv:2504.11456*, 2025.
- 179 [HTL25] Junda He, Christoph Treude, and David Lo. Llm-based multi-agent systems for software
180 engineering: Literature review, vision, and the road ahead. *ACM Transactions on*
181 *Software Engineering and Methodology*, 34(5):1–30, 2025.
- 182 [Lea] Lean Prover Community. Missing undergraduate mathematics in Mathlib. [https://](https://leanprover-community.github.io/undergrad_todo.html)
183 leanprover-community.github.io/undergrad_todo.html. Accessed: Septem-
184 ber 4, 2025.
- 185 [Leb25a] Jiří Lebl. Basic analysis: Introduction to real analysis. [https://github.com/](https://github.com/jirilebl/ra)
186 [jirilebl/ra](https://github.com/jirilebl/ra), 2025. GitHub repository; accessed 2025-09-04.
- 187 [Leb25b] Jiří Lebl. Guide to cultivating complex analysis: Working the complex field. [https://](https://github.com/jirilebl/ca)
188 github.com/jirilebl/ca, 2025. GitHub repository; accessed 2025-09-04.
- 189 [m-a25] m-a-p. Fineleancorpus: A large-scale, high-quality lean 4 formalization dataset. Hugging
190 Face, 2025. NLLean 4 pairs; accessed 2025-09-04.
- 191 [McK25] Benjamin McKay. Topology lecture notes. [https://github.com/Ben-McKay/](https://github.com/Ben-McKay/topology-lecture-notes)
192 [topology-lecture-notes](https://github.com/Ben-McKay/topology-lecture-notes), 2025. GitHub repository; accessed 2025-09-04.
- 193 [PYM⁺25] Zhongyuan Peng, Yifan Yao, Kaijing Ma, Shuyue Guo, Yizhe Li, Yichi Zhang, Chenchen
194 Zhang, Yifan Zhang, Zhouliang Yu, Luming Li, Minghao Liu, Yihang Xia, Jiawei Shen,
195 Yuchen Wu, Yixin Cao, Zhaoxiang Zhang, Wenhao Huang, Jiaheng Liu, and Ge Zhang.
196 Criticlean: Critic-guided reinforcement learning for mathematical formalization. *arXiv*
197 *preprint arXiv:2507.06181*, 2025.
- 198 [RN95] Stuart Russell and Peter Norvig. *Artificial Intelligence: A Modern Approach*. Prentice
199 Hall, 1995. See Chapter 2: Intelligent Agents.

- 200 [SCB⁺24] Noah Shinn, Federico Cassano, Edward Berman, Ashwin Gopinath, Karthik Narasimhan,
201 and Shunyu Yao. Reflexion: Language agents with verbal reinforcement learning. In
202 *International Conference on Learning Representations (ICLR)*, 2024. arXiv:2303.11366.
- 203 [SYA25] Peiyang Song, Kaiyu Yang, and Anima Anandkumar. Lean copilot: Large language
204 models as copilots for theorem proving in lean. 2025.
- 205 [WDL⁺25] Ke Weng, Lun Du, Sirui Li, Wangyue Lu, Haozhe Sun, Hengyu Liu, and Tiancheng
206 Zhang. Autoformalization in the era of large language models: A survey. *arXiv preprint*
207 *arXiv:2505.23486*, 2025.
- 208 [WUL⁺25] Haiming Wang, Mert Unsal, Xiaohan Lin, Mantas Baksys, Junqi Liu, Marco Dos Santos,
209 Flood Sung, Marina Vinyes, Zhenzhe Ying, Zekai Zhu, et al. Kimina-prover preview:
210 Towards large formal reasoning models with reinforcement learning. *arXiv preprint*
211 *arXiv:2504.11354*, 2025.
- 212 [WZJ⁺24] Ruida Wang, Jipeng Zhang, Yizhen Jia, Rui Pan, Shizhe Diao, Renjie Pi, and Tong
213 Zhang. Theoremllama: Transforming general-purpose llms into lean4 experts. *CoRR*,
214 2024.

215 A Technical Appendices and Supplementary Material

216 A.1 Case study on a false positive due to foundational differences

217 We translated the following theorem that one-point compactification produces a topological space:

Let (X, \mathcal{T}) be a Hausdorff topological space. Define $X' := X \cup \{\infty\}$ and

$$\mathcal{T}' := \mathcal{T} \cup \{(X \setminus K) \cup \{\infty\} : K \subseteq X \text{ is compact}\}.$$

Show that \mathcal{T}' is a topology on X' .

218 Here is the compiling Lean file output from the agent:

```
219 import Mathlib
220
221 universe u
222 variable (X : Type u)
223   [TopologicalSpace X] [T2Space X]
224
225 /-- Embed 'X' into 'X PUnit'.
226   Use 'Sum.inr PUnit.unit' as . -/
227 def embed (x : X) : Sum X PUnit :=
228   Sum.inl x
229
230 def infinity : Sum X PUnit :=
231   Sum.inr PUnit.unit
232
233 /-- Opens for the one-point compactification
234   modeled as 'Sum X PUnit'.
235   A set 's' is open iff
236   ( s 'embed 1' s' is open in 'X')
237   ( s K : Set X, 'IsCompact K'
238     'embed 1' s = (Set.univ : Set X) \ K'). -/
239 def isOpen_one_point (s : Set (Sum X PUnit)) : Prop :=
240   (infinity X s IsOpen (embed X 1' s))
241   (infinity X s
242     K : Set X, IsCompact K
243     embed X 1' s =
244
245     (Set.univ : Set X) \ K)
246
247 /-- The topology of the one-point
248   compactification on 'Sum X PUnit'.
249   (Proof omitted.) -/
250 def one_point_compactification_topology :
251   TopologicalSpace
252   (Sum X PUnit) := by
253   sorry
```

254 **Why this is a false positive.** At first glance the translation looks faithful: it models X' as
255 `Sum X PUnit` via `embed`, names the adjoined point `infinity`, and defines the topology by
256 `isOpen_one_point`. However, to inhabit `TopologicalSpace (Sum X PUnit)` one must *spec-*
257 *ify* the topology (i.e., bind `isOpen := isOpen_one_point`) *and prove* the topology axioms (i.e.
258 inhabit `isOpen_univ`, `isOpen_inter`, `isOpen_sUnion`). The line `:= by sorry` leaves both the
259 choice of `isOpen` and the axioms unresolved. For a faithful translation, while we permit `sorry` for
260 proofs we should not for *data*. To fix the translation, we should replace `by sorry` with

```
261 { isOpen      := isOpenOnePoint X,
```

```

262   isOpen_univ   := by sorry,
263   isOpen_inter  := by sorry,
264   isOpen_sUnion := by sorry }

```

265 This issue stems from foundational challenges when converting from informal statements based on
266 ZFC to formal code based on Lean’s dependent type theory. We believe we can enlarge the database
267 of examples by special cases like this, to improve performance.

268 A.2 Case study on a false positive via agent cheating

269 We translated the following theorem from Jiří Lebl’s *Guide to Cultivating Complex Analysis* [Leb25b]:
270 for complex functions, differentiability implies analyticity (i.e., the existence of a Taylor expansion):

If $f: U \rightarrow \mathbb{C}$ is holomorphic on a domain $U \subset \mathbb{C}$, then f is analytic. That is, given $z_0 \in U$, f has a Taylor series, and it converges in the disk $\{z : |z - z_0| < R\}$, where R is the distance to the nearest singularity (possibly $R = \infty$). Furthermore, the sum of the series is f .

271 The ALA without to access to Herald gave the following compiling Lean output:

```

272 import Mathlib
273
274 theorem trivial_nat_eq (x : Nat) : x = x := by
275   rfl

```

276 This code represents a clear tautology, that every natural number is equal to itself; it has no bearing
277 with the original natural language statement. Interestingly, the ALA with access to Herald is giving a
278 correct translation:

```

279 import Mathlib
280 open Complex
281
282 /-- If 'f : U → C' is holomorphic on an open 'U ⊂ C',
283 then 'f' is analytic at each 'z ∈ U'. -/
284 theorem holomorphic_on_analytic
285   (U : Set C) {f : C → C} (hU : IsOpen U) (hf : DifferentiableOn f U)
286   (z0 : C) (hz0 : z0 ∈ U) :
287   AnalyticAt C f z0 := by
288   sorry

```

289 **Why this is happening.** We can compare logs in the outputs folder with the results from access to
290 Herald and without (our fine-tuned LLM). These logs indicate that the ALA without Herald made
291 an incorrect call to "path": "test.lean", erasing its previous attempts. These types of failures, of
292 reducing to a degenerate proof, were rare (we found around 7 in the experiment where the ALA does
293 not have access to Herald). We did not include the evaluator of faithfulness into the feedback loop;
294 this might have kept the Agent on track.

295 A.3 Statistical discussion

296 **Time-to-first-success analysis.** We analyze *time to first successful compile*, measured in the discrete
297 unit of *number of tool calls*. Each run belongs to one of two conditions: **ALA: Agent with access**
298 **to all tools.** or a **Agent with access to Herald-only** condition. We fit a Cox proportional hazards
299 model with a single binary covariate for condition (ALA= 1, Herald-only= 0), *stratified by theorem*
300 so that each theorem has its own baseline hazard. Runs without a success by the administrative limit
301 $K = 24$ calls are *right-censored at $t = 24$* ; events that occur at $t = 24$ are counted as events (not
302 censored). Because time is recorded in integer calls, we handle *tied event times* using the *Efron*
303 method. Hazard ratios (HR) > 1 indicate faster success (fewer calls on average) for ALA relative to
304 Herald-only. Model diagnostics included checks of the proportional-hazards assumption (global and
305 covariate-specific Schoenfeld residual tests/plots). We report the number of strata (theorems), total
306 runs, number of events, and the censoring proportion. Table 2 summarizes the fitted model.

Table 2: Cox proportional hazards regression results. HR = hazard ratio = $\exp(\text{Coef})$.

Term	Coef	SE(Coef)	z	p	HR	HR 95% L	HR 95% U
Fine-tuned LLM	0.111	0.102	1.087	0.277	1.117	0.915	1.364

307 In a theorem-stratified Cox model, the fine-tuned LLM showed a higher hazard of first successful
 308 compile (HR = 1.117, 95% CI [0.915, 1.364], $z = 1.087$, $p = 0.277$), implying an estimated 11.7%
 309 faster per-call success rate but with uncertainty spanning from 8.5% slower to 36.4% faster; thus the
 310 effect is not statistically significant across all possible number of tool calls.

311 A.4 Agents workflows

312 We wrote the system prompt (see the Appendix A.6) to suggest one possible workflow to better
 313 generate high quality data, by give outline of translation process and contingency plan to handle
 314 unsuccessful translations. The agent has `max_step` times to use the tools, the agent will a return
 315 JSON flag if its did the writing of Lean 4 code into disk, and use `run_lean_tool` to verify if the
 316 Lean 4 code compiles. There might be cases that, during the last step the agent write a code but have
 317 not had chance to using `run_lean_tool` to evaluate, so after the agent finishing processing all input,
 318 we re-evaluate those cases whose status is `max_step_reached`. After agent running, we have
 319 a csv file, which columns are name, step, status, passed, `nl_statement`, `lean4_code`,
 320 then we re-evaluate those status:`max_step_reached` to fix the potential false negative.

321 For each row, we read the pair (`nl_statement`, `lean4_code`), send to LLM judge (GPT-5 with
 322 reasoning="effort": "medium",) to evaluates whether the Lean 4 code faithfully represents the
 323 natural language statement. We then augment the CSV with three new columns:

- 324 • `validate_score`: a base-10 numerical score indicating the degree of faithfulness,
- 325 • `validate_reason`: a free-form textual rationale explaining why the translation is (or is
 326 not) valid,
- 327 • `equivalent`: a Boolean flag (True/False) specifying whether the natural-language state-
 328 ment and Lean 4 code are judged equivalent. In our rubrics, True only if the score is
 329 10.

330 The below is the pseudo algorithm description:

Algorithm 1 Lean4 translation agent (controller + post-processing)

Require: statement `nl_statement`, file path `p`, tools \mathcal{T} , step limit S_{\max}

- 1: $History \leftarrow [(\text{system}, \pi), (\text{user}, \text{"Translate " } x \text{ and save to } p)]$ $\triangleright \pi$: system policy
- 2: **for** $s = 1$ **to** S_{\max} **do**
- 3: $resp \leftarrow \text{Model}(History, \mathcal{T})$ \triangleright returns either content or a single tool call
- 4: **if** "status": "success" $\in resp.content$ **then**
- 5: **return** SUCCESS \triangleright explicit success token
- 6: **end if**
- 7: **if** $resp.tool_calls \neq \emptyset$ **then**
- 8: $(toolName, argument) \leftarrow$ first tool and its JSON args
- 9: $result \leftarrow tool.run(argument)$
- 10: $History \leftarrow History \cup [(tool, result)]$
- 11: **if** `tool = lean4_repl_runner` **and** `result.repl_pass = 1` **then**
- 12: **return** SUCCESS \triangleright auto-stop on REPL pass
- 13: **end if**
- 14: **else**
- 15: $History \leftarrow History \cup [(assistant, resp.content)]$
- 16: **end if**
- 17: **end for**
- 18: **return** MAXSTPREACHED \triangleright may have written code but not verified

Algorithm 2 Post-processing: REPL re-check and LLM judging

Require: CSV with columns name, step, status, passed, nl_statement, lean4_code

```
1: for each row  $r \in \mathcal{C}$  do
2:   if row.status = max_step_reached then
3:     result  $\leftarrow$  lean4_repl_runner(row.lean4_code)
4:     update row.passed  $\leftarrow$  (row.repl_pass = 1)
5:   end if
6: end for
7: for each row  $r \in \mathcal{C}$  do
8:    $(\hat{y}, \rho) \leftarrow$  GPT-5-JUDGE(row.nl_statement, row.lean4_code)
9:   r.judge_score  $\leftarrow$   $\hat{y}$ ; r.judge_rationale  $\leftarrow$   $\rho$ 
10: end for
11: return updated  $\mathcal{C}$ 
```

331 A.5 Prompt for evaluating correctness of translation

332 Compiling Lean 4 code does not guarantee that the translation is correct; it can pass for the reasons
333 outlined in Appendices A.1 and A.2. Following Herald Translator [GWJ⁺25], we employ an LLM
334 judge to evaluate faithfulness. We use the following 1-shot CoT prompt with GPT-5 (reasoning mode:
335 medium) for evaluating faithfulness.

336 You are an expert in Lean 4, mathlib, and mathematics. You are an
337 auditor with guidelines.

338

339 Instructions:

340 Your input is (A) compiling Lean 4 code and (B) a natural-language
341 statement. Decide whether (A) faithfully formalizes (B). Do not use
342 proof quality; only check statement fidelity.

343

344 Think step by step:

- 345 1) Translate each line of the Lean 4 code into plain language. Check
346 if it is sensible and on track.
347 2) Then decide if the whole Lean statement is faithful to the original.
348 3) Final check: are the two math statements the same or different?
349 Point out any differences precisely.

350

351 Guidelines:

- 352 1) It must be a legitimate, faithful translation to pass. Small
353 formalization differences are fine. Since the code compiles, assume
354 referenced names exist in mathlib.
355 2) Prefer current/standard mathlib terms; ad-hoc encodings can be a
356 red flag if they change meaning.
357 3) If the Lean code introduces auxiliary definitions (beyond the final
358 theorem/definition), they must not be vacuous. If any auxiliary
359 definition is vacuous (e.g., ‘:= True’, ‘:= none’, or filled with
360 ‘sorry’ where data is required), the translation fails. The aux
361 definition must describe what it is trying to say.
362 4) Only if each auxiliary definition is legitimate and the final Lean
363 statement matches the original in mathematical meaning does it pass.
364 Do not penalize harmless formal phrasing differences.
365 5) If it is a near pass, assess whether the Lean 4 statement is a good
366 formalization of the original. Slight specialization/generalization
367 is acceptable if no substantive error is introduced.

368

369 After you finish your reasoning:

370 Assign a Grade $\{0, \dots, 10\}$. Use this rubric:

371 0: completely unrelated

372 3: vacuous aux defs and even fixing them would not make it faithful

```

373 6: vacuous aux defs, but fixing them would make it faithful
374 9: almost the same but still not faithful
375 10: faithful
376
377 Also output:
378 - COT inside "### BEGIN THOUGHT" / "### END THOUGHT".
379 - Faithfulness (binary) immediately after "### FAITHFUL SCORE"
380   where 0 = not faithful, 1 = faithful.
381 - The numeric grade immediately after "### GRADE".
382
383 ---
384
385 ### Example
386
387 Here is the Lean 4 code:
388 ```lean
389 import Mathlib
390
391 universe u v
392 variables {X : Type u} {Y : Type v}
393   [TopologicalSpace X] [TopologicalSpace Y]
394
395 /-- Placeholder for a covering map. -/
396 def CoveringMap (p : X → Y) : Prop := True
397
398 /-- Placeholder: U is evenly covered by p. -/
399 def evenly_covered (p : X → Y) (U : Set Y) : Prop := True
400
401 /-- Number of sheets (none = ). -/
402 def num_sheets (p : X → Y) (U : Set Y) : Option Nat := none
403
404 /-- Placeholder for path connectedness. -/
405 def PathConnected (Y : Type v) [TopologicalSpace Y] : Prop := True
406
407 namespace Covering
408
409 theorem sheets_equal_on_overlap {p : X → Y} (hp : CoveringMap p)
410   {U V : Set Y} (heU : evenly_covered p U) (heV : evenly_covered p V)
411   (hnonempty : (U ∩ V).Nonempty) :
412   num_sheets p U = num_sheets p V := by sorry
413
414 theorem covering_map_n_to_one_of_path_connected {p : X → Y}
415   (hp : CoveringMap p) (hpath : PathConnected Y) :
416   (n : Option Nat), (y : Y), (U : Set Y),
417   y ∈ U → IsOpen U → evenly_covered p U → num_sheets p U = n := by
418   sorry
419
420 end Covering

```

421 Note, the grade is an artificial value not used in the final analysis. It was a book-keeping device for
422 us to keep track of uncertainty. A grade 0 happens usually only when the proof degenerate into a
423 triviality as in Appendix A.2. A grade 9 sometimes happens for false negatives (correct translated
424 code that was judged too harshly by this evaluator). If the translation is deemed faithful, the LLM
425 outputs a faithful score of 1; otherwise it outputs 0.

426 A.6 Agent Prompts

427 We provide four system prompts, one for each ALA configuration we tested: (1) full ALA (all tools,
428 including the specialist LLM Herald), (2) ALA without the specialist LLM, (3) specialist LLM only,

429 and (4) ALA without REPL and without the specialist LLM. The exact prompt strings and tool-call
430 templates are available in the anonymized code repository.

431 **ALA with access to tools including the specialist LLM**

```
432 You are an expert Lean4 programmer-agent.
433 Translate the NL math statement into ONE Lean4
434 declaration that compiles with Mathlib.
435 Translation only - **not a full proof**.
436
437 Always call 'lean4_translation' FIRST. Then write
438 to disk and verify with 'lean4_repl_runner'.
439 Pass = '1' (compiles); Fail = '0' (does not).
440
441 When translating:
442 - add 'import Mathlib' at the top
443 - end the decl with ':= by sorry' (no proof)
444
445 ## Process
446 1) (Optional, once) 'lean_retrieval' for an example.
447 2) Translate: use 'lean4_translation' or draft manually
448    (always end with ':= by sorry').
449 3) Write & verify: 'lean_write_file' → 'lean4_repl_runner'.
450 4) If 'repl_pass = 1' → respond '{ "status": "success" }',
451    else revise and retry.
452
453 ## Contingency (errors)
454 - Unknown names → 'lean_check_theorem'.
455 - Syntax/tactics → 'search_online'.
456 - Re-test → 'lean4_repl_runner'; iterate.
457
458 ## Naming (Lean4/mathlib)
459 - Types/Props: PascalCase
460   e.g., 'IsSimpleGroup', 'IsCyclic', 'Nat.Prime'
461 - Lemmas/Functions: snake_case
462   e.g., 'Nat.add_comm', 'List.map'
463 - Be specific: prefer
464   'Sylow.exists_subgroup_card_pow_prime'
465   over vague labels like "Sylow Theorem".
```

466 **ALA with access to tools except the specialist LLM**

```
467 You are an expert Lean4 programmer-agent. Translate the given
468 natural-language statement into a single Lean4 declaration that
469 compiles with Mathlib. Translation only, not a full proof.
470
471 Draft the statement yourself (no specialist translator). Write
472 it to disk and verify with 'lean4_repl_runner'. Pass = 1, fail = 0.
473
474 When translating, import 'Mathlib' at the top and end with
475 ':= by sorry' (no proof).
476
477 Process
478 1) (Optional, once) 'lean_retrieval' for an example.
479 2) 'lean_write_file' → 'lean4_repl_runner'.
480 3) If 'repl_pass = 1', respond '{ "status": "success" }';
481    else revise and retry.
482
483 Errors
```

484 - Unknown names: 'lean_check_theorem'.
 485 - Syntax/tactics: 'search_online'.
 486 - Re-test with 'lean4_repl_runner' and iterate.
 487
 488 Naming (Lean4/Mathlib)
 489 - Types/Props: PascalCase (e.g., 'IsSimpleGroup', 'Nat.Prime')
 490 - Lemmas/Fns: snake_case (e.g., 'Nat.add_comm', 'List.map')
 491 - Prefer specific names (e.g., 'Sylow.exists_subgroup_card_pow_prime')

492 **ALA without access to any other tools except the specialist LLM**

493 You are an expert Lean4 programmer-agent.
 494 Translate the given NL statement into ONE
 495 Lean4 declaration that compiles with Mathlib.
 496 Translation only - not a full proof.
 497
 498 Use 'lean4_translation' to draft the declaration,
 499 then write it to disk with 'lean_write_file'.
 500
 501 When translating:
 502 - add 'import Mathlib' at the top
 503 - end the decl with ':= by sorry' (no proof)
 504
 505 When finished, respond with:
 506 { "status": "success" }

507 **ALA without REPL and without the specialist LLM**

508 You are an expert Lean4 programmer-agent. Your mission is to translate
 509 the given natural-language statement into a single Lean4 declaration.
 510 Your goal is translation only, not a full proof.
 511
 512 After generating the code, write it to disk with 'lean_write_file'.
 513
 514 When translating, import 'Mathlib' at the top and end the declaration
 515 with ':= by sorry' (no proof).
 516
 517 **## Tools**
 518 - 'check_theorem_tool': check existence / canonical names.
 519 - 'lean_write_file': write code to disk.
 520 - 'lean4_translation': draft a declaration (no proof). You may use it,
 521 but verify syntax/names with other tools; do not rely on it alone.
 522 - 'lean_retrieval': fetch similar (NL, Lean) example pairs.
 523
 524 **## Naming**
 525 1. Types/Props: PascalCase (e.g., 'IsSimpleGroup', 'Nat.Prime').
 526 2. Lemmas/Functions: snake_case (e.g., 'Nat.add_comm', 'List.map').
 527 3. Be specific: prefer 'Sylow.exists_subgroup_card_pow_prime'.
 528 4. Confirm names with 'check_theorem_tool'.
 529
 530 Respond with: '{ "status": "success" }' once the translation is written.

531 **NeurIPS Paper Checklist**

532 **1. Claims**

533 Question: Do the main claims made in the abstract and introduction accurately reflect the
534 paper’s contributions and scope?

535 Answer: [Yes]

536 Justification: We introduce the ALA framework (Sec. 3), a 400-example upper-
537 division/graduate NL dataset (Sec. 4), and an empirical evaluation on four Lean 4 domains
538 (Secs. 5, 6). The scope is bounded to the specified models, Mathlib library, and a 24-call
539 budget.

540 Guidelines:

541 **2. Limitations**

542 Question: Does the paper discuss the limitations of the work performed by the authors?

543 Answer: [Yes]

544 Justification: We discuss potential web contamination of the evaluation set, possible false
545 positives/negatives from the LLM-based faithfulness evaluator, sensitivity to the agent
546 prompt, dataset size, and RAG pool, runtime constraints from the REPL tool, and a post-
547 processing gap (only max-step cases are rechecked). See Sec. 6.1.

548 **3. Theory assumptions and proofs**

549 Question: For each theoretical result, does the paper provide the full set of assumptions and
550 a complete (and correct) proof?

551 Answer: [NA]

552 Justification: The paper introduces a system and reports empirical results. It does not
553 present new theorems or proofs; it does include a case study of generated formal code in the
554 appendix A.1)

555 **4. Experimental result reproducibility**

556 Question: Does the paper fully disclose all the information needed to reproduce the main ex-
557 perimental results of the paper to the extent that it affects the main claims and/or conclusions
558 of the paper (regardless of whether the code and data are provided or not)?

559 Answer: [Yes]

560 Justification: Section 3 specifies the pipeline (orchestrator + tools) and the controller/loops
561 (Algorithms 1, 2). Section 5 explains datasets , model choices (including Herald Translator),
562 the 24-call limit on tools, and the success criterion used to compute Pass@k.

563 **5. Open access to data and code**

564 Question: Does the paper provide open access to the data and code, with sufficient instruc-
565 tions to faithfully reproduce the main experimental results, as described in supplemental
566 material?

567 Answer: [Yes]

568 Justification: Our anonymized GitHub repo contains the runnable pipeline (code, tools,
569 configs, inputs, instructions, and requirements). We use Lean’s REPL feedback tool, a
570 secondary evaluator LLM, Herald Translator, and a RAG database of 500 examples.

571 Guidelines:

- 572 • The answer NA means that paper does not include experiments requiring code.
- 573 • Please see the NeuroIPS code and data submission guidelines ([https://nips.cc/
574 public/guides/CodeSubmissionPolicy](https://nips.cc/public/guides/CodeSubmissionPolicy)) for more details.
- 575 • While we encourage the release of code and data, we understand that this might not be
576 possible, so “No” is an acceptable answer. Papers cannot be rejected simply for not
577 including code, unless this is central to the contribution (e.g., for a new open-source
578 benchmark).
- 579 • The instructions should contain the exact command and environment needed to run to
580 reproduce the results. See the NeurIPS code and data submission guidelines ([https://
581 //nips.cc/public/guides/CodeSubmissionPolicy](https://nips.cc/public/guides/CodeSubmissionPolicy)) for more details.

- 582 • The authors should provide instructions on data access and preparation, including how
583 to access the raw data, preprocessed data, intermediate data, and generated data, etc.
- 584 • The authors should provide scripts to reproduce all experimental results for the new
585 proposed method and baselines. If only a subset of experiments are reproducible, they
586 should state which ones are omitted from the script and why.
- 587 • At submission time, to preserve anonymity, the authors should release anonymized
588 versions (if applicable).
- 589 • Providing as much information as possible in supplemental material (appended to the
590 paper) is recommended, but including URLs to data and code is permitted.

591 6. Experimental setting/details

592 Question: Does the paper specify all the training and test details (e.g., data splits, hyper-
593 parameters, how they were chosen, type of optimizer, etc.) necessary to understand the
594 results?

595 Answer: [Yes]

596 Justification: The paper specifies the evaluation dataset (Secs. 4, 5); the exact models/tools
597 (generalist LLM + Herald; REPL, RAG examples, Mathlib check, and web search via the
598 Serper API) are available in the anonymized code repository. We use Lean 4 (4.15.0), a
599 24-call budget, and fixed seeds/prompts/configs (see Appendix A.5, A.6).

600 Guidelines:

- 601 • The answer NA means that the paper does not include experiments.
- 602 • The experimental setting should be presented in the core of the paper to a level of detail
603 that is necessary to appreciate the results and make sense of them.
- 604 • The full details can be provided either with the code, in appendix, or as supplemental
605 material.

606 7. Experiment statistical significance

607 Question: Does the paper report error bars suitably and correctly defined or other appropriate
608 information about the statistical significance of the experiments?

609 Answer: [Yes]

610 Justification: For Pass@ k we plot 95% confidence intervals per k as binomial CIs over
611 $N=400$ problems. These appear as the error bars in our figures.

612 Guidelines:

- 613 • The answer NA means that the paper does not include experiments.
- 614 • The authors should answer "Yes" if the results are accompanied by error bars, confi-
615 dence intervals, or statistical significance tests, at least for the experiments that support
616 the main claims of the paper.
- 617 • The factors of variability that the error bars are capturing should be clearly stated (for
618 example, train/test split, initialization, random drawing of some parameter, or overall
619 run with given experimental conditions).
- 620 • The method for calculating the error bars should be explained (closed form formula,
621 call to a library function, bootstrap, etc.)
- 622 • The assumptions made should be given (e.g., Normally distributed errors).
- 623 • It should be clear whether the error bar is the standard deviation or the standard error
624 of the mean.
- 625 • It is OK to report 1-sigma error bars, but one should state it. The authors should
626 preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis
627 of Normality of errors is not verified.
- 628 • For asymmetric distributions, the authors should be careful not to show in tables or
629 figures symmetric error bars that would yield results that are out of range (e.g. negative
630 error rates).
- 631 • If error bars are reported in tables or plots, The authors should explain in the text how
632 they were calculated and reference the corresponding figures or tables in the text.

633 8. Experiments compute resources

634 Question: For each experiment, does the paper provide sufficient information on the com-
635 puter resources (type of compute workers, memory, time of execution) needed to reproduce
636 the experiments?

637 Answer: [Yes]

638 Justification: For the tool `herald_translator_tool`, it's take about 14GB GPU memory
639 to load the model. We used a virtual machine on Goole Cloud with CPU `a2-highgpu-1g`
640 (12 vCPUs, 85 GB Memory) with GPU NVIDIA A100 40GB to serve the model using
641 `vllm`.

642 For the agent running, we are running on the script over a Apple M3 Pro with 18GB
643 memory. It take 1 to 1.5 hour to go over the 400 theorems depending on the selection of
644 tools. Details discussed in the limitation.

645 For the judgment script, since we asked the model GPT-5 with thinking mode `medium`,
646 the average time for generating score and reasoning is around 20 mins and cost
647 around \$14 for go over 400 (`nl_statement`, `Lean4_code`), the log file is under
648 `all_experiments_csv/record.txt`.

649 Guidelines:

- 650 • The answer NA means that the paper does not include experiments.
- 651 • The paper should indicate the type of compute workers CPU or GPU, internal cluster,
652 or cloud provider, including relevant memory and storage.
- 653 • The paper should provide the amount of compute required for each of the individual
654 experimental runs as well as estimate the total compute.
- 655 • The paper should disclose whether the full research project required more compute
656 than the experiments reported in the paper (e.g., preliminary or failed experiments that
657 didn't make it into the paper).

658 9. Code of ethics

659 Question: Does the research conducted in the paper conform, in every respect, with the
660 NeurIPS Code of Ethics <https://neurips.cc/public/EthicsGuidelines>?

661 Answer: [Yes]

662 Justification: We use only openly licensed educational materials and tools made by the
663 Lean 4 community; no personal or sensitive data are involved. For our dataset, we use Lebl,
664 McKay, and Doty's repositories [Leb25a, Leb25b, McK25, Dot25]); we respect upstream
665 licenses and terms of use.

666 Guidelines:

- 667 • The answer NA means that the authors have not reviewed the NeurIPS Code of Ethics.
- 668 • If the authors answer No, they should explain the special circumstances that require a
669 deviation from the Code of Ethics.
- 670 • The authors should make sure to preserve anonymity (e.g., if there is a special consid-
671 eration due to laws or regulations in their jurisdiction).

672 10. Broader impacts

673 Question: Does the paper discuss both potential positive societal impacts and negative
674 societal impacts of the work performed?

675 Answer: [Yes]

676 Justification: Positive impacts include improving formalization for these target areas (upper-
677 division undergraduate and graduate mathematics); risks include unfaithful translations. We
678 mitigate this via an LLM evaluator but mistakes can still happen.

679 Guidelines:

- 680 • The answer NA means that there is no societal impact of the work performed.
- 681 • If the authors answer NA or No, they should explain why their work has no societal
682 impact or why the paper does not address societal impact.
- 683 • Examples of negative societal impacts include potential malicious or unintended uses
684 (e.g., disinformation, generating fake profiles, surveillance), fairness considerations
685 (e.g., deployment of technologies that could make decisions that unfairly impact specific
686 groups), privacy considerations, and security considerations.

- 687 • The conference expects that many papers will be foundational research and not tied
688 to particular applications, let alone deployments. However, if there is a direct path to
689 any negative applications, the authors should point it out. For example, it is legitimate
690 to point out that an improvement in the quality of generative models could be used to
691 generate deepfakes for disinformation. On the other hand, it is not needed to point out
692 that a generic algorithm for optimizing neural networks could enable people to train
693 models that generate Deepfakes faster.
- 694 • The authors should consider possible harms that could arise when the technology is
695 being used as intended and functioning correctly, harms that could arise when the
696 technology is being used as intended but gives incorrect results, and harms following
697 from (intentional or unintentional) misuse of the technology.
- 698 • If there are negative societal impacts, the authors could also discuss possible mitigation
699 strategies (e.g., gated release of models, providing defenses in addition to attacks,
700 mechanisms for monitoring misuse, mechanisms to monitor how a system learns from
701 feedback over time, improving the efficiency and accessibility of ML).

702 11. Safeguards

703 Question: Does the paper describe safeguards that have been put in place for responsible
704 release of data or models that have a high risk for misuse (e.g., pretrained language models,
705 image generators, or scraped datasets)?

706 Answer: [NA]

707 Justification: We do not release high-risk assets; our artifacts are code and a small text
708 dataset of mathematical statements from openly licensed academic sources.

709 Guidelines:

- 710 • The answer NA means that the paper poses no such risks.
- 711 • Released models that have a high risk for misuse or dual-use should be released with
712 necessary safeguards to allow for controlled use of the model, for example by requiring
713 that users adhere to usage guidelines or restrictions to access the model or implementing
714 safety filters.
- 715 • Datasets that have been scraped from the Internet could pose safety risks. The authors
716 should describe how they avoided releasing unsafe images.
- 717 • We recognize that providing effective safeguards is challenging, and many papers do
718 not require this, but we encourage authors to take this into account and make a best
719 faith effort.

720 12. Licenses for existing assets

721 Question: Are the creators or original owners of assets (e.g., code, data, models), used in
722 the paper, properly credited and are the license and terms of use explicitly mentioned and
723 properly respected?

724 Answer: [Yes]

725 Justification: We credit all third-party assets in the paper; here are their licenses:
726 Lean 4/Mathlib/REPL tool (Apache-2.0), Herald Translator (Apache-2.0; weights not
727 redistributed), Lebl notes (CC BY-SA / CC BY-NC-SA), McKay topology notes (GPL-3.0),
728 and Doty algebra notes (MIT for code; CC BY 4.0 for text). Their terms are stated and
729 respected.

730 Guidelines:

- 731 • The answer NA means that the paper does not use existing assets.
- 732 • The authors should cite the original paper that produced the code package or dataset.
- 733 • The authors should state which version of the asset is used and, if possible, include a
734 URL.
- 735 • The name of the license (e.g., CC-BY 4.0) should be included for each asset.
- 736 • For scraped data from a particular source (e.g., website), the copyright and terms of
737 service of that source should be provided.
- 738 • If assets are released, the license, copyright information, and terms of use in the
739 package should be provided. For popular datasets, paperswithcode.com/datasets

740 has curated licenses for some datasets. Their licensing guide can help determine the
741 license of a dataset.

- 742 • For existing datasets that are re-packaged, both the original license and the license of
743 the derived asset (if it has changed) should be provided.
- 744 • If this information is not available online, the authors are encouraged to reach out to
745 the asset’s creators.

746 13. New assets

747 Question: Are new assets introduced in the paper well documented and is the documentation
748 provided alongside the assets?

749 Answer: [Yes]

750 Justification: We introduce an agent and a 400-example dataset; both are documented in the
751 anonymized GitHub (see the README and LICENSES).

752 Guidelines:

- 753 • The answer NA means that the paper does not release new assets.
- 754 • Researchers should communicate the details of the dataset/code/model as part of their
755 submissions via structured templates. This includes details about training, license,
756 limitations, etc.
- 757 • The paper should discuss whether and how consent was obtained from people whose
758 asset is used.
- 759 • At submission time, remember to anonymize your assets (if applicable). You can either
760 create an anonymized URL or include an anonymized zip file.

761 14. Crowdsourcing and research with human subjects

762 Question: For crowdsourcing experiments and research with human subjects, does the paper
763 include the full text of instructions given to participants and screenshots, if applicable, as
764 well as details about compensation (if any)?

765 Answer: [NA]

766 Justification: The paper does not involve crowdsourcing nor research with human subjects.

767 Guidelines:

- 768 • The answer NA means that the paper does not involve crowdsourcing nor research with
769 human subjects.
- 770 • Including this information in the supplemental material is fine, but if the main contribu-
771 tion of the paper involves human subjects, then as much detail as possible should be
772 included in the main paper.
- 773 • According to the NeurIPS Code of Ethics, workers involved in data collection, curation,
774 or other labor should be paid at least the minimum wage in the country of the data
775 collector.

776 15. Institutional review board (IRB) approvals or equivalent for research with human 777 subjects

778 Question: Does the paper describe potential risks incurred by study participants, whether
779 such risks were disclosed to the subjects, and whether Institutional Review Board (IRB)
780 approvals (or an equivalent approval/review based on the requirements of your country or
781 institution) were obtained?

782 Answer: [NA]

783 Justification: The paper does not involve crowdsourcing nor research with human subjects.

784 Guidelines:

- 785 • The answer NA means that the paper does not involve crowdsourcing nor research with
786 human subjects.
- 787 • Depending on the country in which research is conducted, IRB approval (or equivalent)
788 may be required for any human subjects research. If you obtained IRB approval, you
789 should clearly state this in the paper.

790
791
792
793
794
795
796
797
798
799
800
801
802
803
804
805
806
807
808
809
810

- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. Declaration of LLM usage

Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does not impact the core methodology, scientific rigorosity, or originality of the research, declaration is not required.

Answer: [Yes]

Justification: LLMs are core to our method: a generalist LLM orchestrates tool calls (REPL feedback, RAG retrieval, web search, Mathlib checks); it invokes a Lean 4-specialized translator (Herald Translator) and a secondary LLM evaluates faithfulness. Outside of this core pipeline, an LLM assisted dataset selection, picking 400 problems based on diversity/length (Secs. 4), and 500 RAG examples.

Guidelines:

- The answer NA means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
- Please refer to our LLM policy (<https://neurips.cc/Conferences/2025/LLM>) for what should or should not be described.