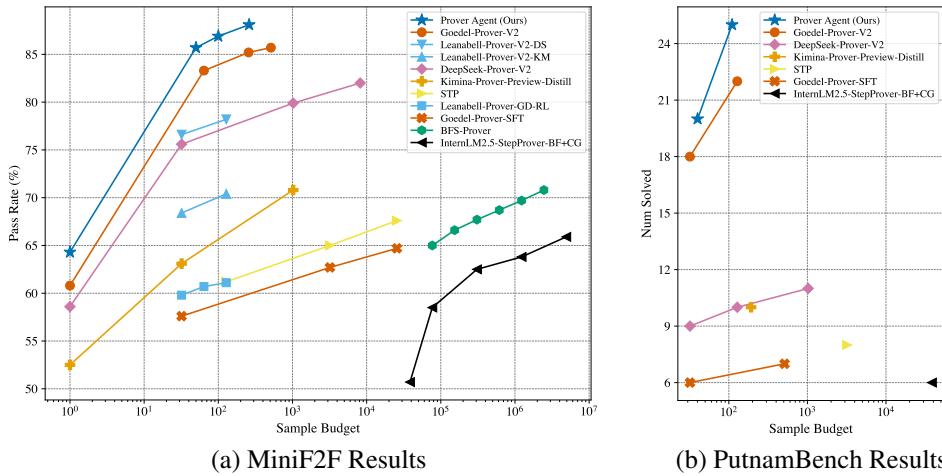


PROVER AGENT: AN AGENT-BASED FRAMEWORK FOR FORMAL MATHEMATICAL PROOFS

000
001
002
003
004
005 **Anonymous authors**
006 Paper under double-blind review
007
008
009
010

ABSTRACT

011 We present Prover Agent, a novel AI agent for automated theorem proving that
012 integrates large language models (LLMs) with a formal proof assistant, Lean. Prover
013 Agent coordinates an informal reasoning LLM, a formal prover model, and feed-
014 back from Lean while also generating auxiliary lemmas. These auxiliary lemmas
015 are not limited to subgoals in the formal proof but can also include special cases or
016 potentially useful facts derived from the assumptions, which help in discovering
017 a viable proof strategy. It achieves an 88.1% success rate on the MiniF2F bench-
018 mark and solves 25 problems on the PutnamBench with a smaller sample budget
019 than previous approaches, establishing a new state-of-the-art on both benchmarks
020 among methods using small language models (SLMs). We also present theoretical
021 analyses and case studies that illustrate how these generated lemmas contribute to
022 solving challenging problems.
023



038 Figure 1: Comparison of theorem-proving performance on MiniF2F (Zheng et al., 2022) and Put-
039 namBench (Tsoukalas et al., 2024b) among methods using SLMs. On both benchmarks, our ap-
040 proach achieves a higher success rate with a smaller sample budget, establishing a new state-of-the-
041 art at this scale.

1 INTRODUCTION

045 Recent advances in the reasoning capabilities of large language models (LLMs) have driven remark-
046 able progress across many areas of artificial intelligence, including mathematical theorem proving
047 and problem solving (OpenAI, 2024; DeepSeek-AI, 2025; Yang et al., 2025a; Lewkowycz et al.,
048 2022). However, LLMs are prone to errors and hallucinations that can undermine their reliability (Ji
049 et al., 2023; Huang et al., 2025; Xu et al., 2025). Inference-time scaling techniques such as chain-
050 of-thought have greatly enhanced their reasoning performance by allowing models to reflect on and
051 correct faulty reasoning steps (Wei et al., 2022). Nonetheless, eliminating mistakes entirely remains
052 challenging, especially for more difficult problems (Wei et al., 2022; Zeng et al., 2025).

053 Formal proof assistants such as Lean (Moura & Ullrich, 2021), The Rocq Prover (previously known
054 as Coq) (Barras et al., 1999), and Isabelle (Paulson, 1994) rigorously verify by computer that every

054 inference step in mathematical proofs written in their respective languages is correct, based on the
 055 Curry–Howard correspondence. This helps mathematicians verify the correctness of proofs. Here,
 056 no errors, omissions of detail, implicit assumptions, or ambiguities are permitted. However, working
 057 with formal proof assistants typically requires painstaking manual effort and meticulous detail. As
 058 a result, automating mathematical theorem proving has long been a grand challenge in artificial
 059 intelligence and formal methods (Newell & Simon, 1956; Irving et al., 2016; Polu & Sutskever,
 060 2020; Jiang et al., 2023; Lu et al., 2023).

061 Consequently, formal theorem proving with LLMs has become increasingly important in recent
 062 years, leading to a growing body of research in this area (Wang et al., 2024b; Wu et al., 2024a; Xin
 063 et al., 2025b; Li et al., 2025; Xin et al., 2025a; Dong & Ma, 2025; Lin et al., 2025b; Zhang et al.,
 064 2025; Wang et al., 2025; Ren et al., 2025; Ji et al., 2025; Lin et al., 2025c; Cao et al., 2025; Zhou
 065 et al., 2025; Chen et al., 2025). This not only provides a way to guarantee the correctness of mathematical
 066 reasoning by LLMs, but also marks a major breakthrough in automated theorem proving. A
 067 key point is the complementary strengths of LLMs and formal proof assistants: LLMs excel in reasoning
 068 and generation but may produce errors and lack guarantees of correctness, whereas formal
 069 proof assistants, such as Lean, possess perfect verification capabilities grounded in mathematical
 070 logic but are not generative.

071 Yet, significant hurdles remain in bridging informal reasoning and formal proving (Yang et al.,
 072 2025b). For instance, prompting o3-mini (OpenAI, 2025) to directly generate a complete Lean
 073 proof for a competition-level problem succeeds in only 6.0% of cases in a single attempt, despite its
 074 strong performance on competition-level mathematical reasoning in natural language (Yousefzadeh
 075 & Cao, 2025). Even when fine-tuned on mathematical data, trained with reinforcement learning, or
 076 allowed chain-of-thought, purely neural approaches fail to produce correct formal proofs, and their
 077 formal proving capabilities still lag far behind their informal reasoning skills in natural language.

078 To bridge this gap between informal reasoning and formal proving, we propose a novel agent frame-
 079 work (**Prover Agent**) that coordinates an informal reasoning LLM, a formal prover model, and
 080 the Lean verification system. To tackle difficult problems that cannot be solved directly, the agent
 081 generates auxiliary lemmas to assist in discovering a viable proof strategy. These lemmas are not
 082 limited to subgoals that can be directly inserted into a formal proof, but may also include special
 083 cases or potentially useful facts derived from the assumptions. Such lemmas are particularly useful
 084 when the overall proof strategy is not apparent from the outset, as they help in constructing a vi-
 085 able plan. It achieves an 88.1% success rate on the MiniF2F benchmark (Zheng et al., 2022) and
 086 solves 25 problems on the PutnamBench (Tsoukalas et al., 2024b), establishing a new state-of-the-
 087 art on both benchmarks among methods using small language models (SLMs). Notably, it uses only
 088 SLMs with a smaller sample budget and a smaller token budget than previous high-performing ap-
 089 proaches, making it much more efficient in terms of inference-time cost. Furthermore, we provide
 090 both a theoretical analysis and a case study to demonstrate the effectiveness of our agent’s approach
 091 to generating auxiliary lemmas.

092 Our contributions are summarized as follows:

- 093 • **Coordination of Informal and Formal Reasoning with Lean Feedback:** Our agent com-
 094 bines an informal LLM and a formal prover under Lean’s verification. The LLM produces
 095 natural language reasoning and lemmas, which the prover formalizes and Lean checks.
 096 Errors detected by Lean are immediately fed back, enabling iterative refinement of con-
 097 structed proofs.
- 098 • **Auxiliary Lemma Generation for Strategy Discovery:** For challenging problems that
 099 cannot be solved directly, our agent generates auxiliary lemmas, such as special cases,
 100 potentially useful facts, or hypothesis-driven conjectures, which are then formally proved.
 101 By reconsidering the overall proof in light of the verified lemmas, the system uncovers
 102 viable proof strategies even when the solution path is not apparent at first.
- 103 • **State-of-the-Art Theorem-Proving Performance:** On the MiniF2F benchmark (Zheng
 104 et al., 2022), a standard benchmark for formal theorem proving that consists of 488
 105 problems drawn from mathematics Olympiads and advanced mathematics, our agent
 106 achieves 88.1% pass rate, establishing a new state-of-the-art among methods using SLMs.
 107 Furthermore, our agent successfully solves 25 problems on more challenging Putnam-
 108 Bench (Tsoukalas et al., 2024b), also achieving state-of-the-art performance among SLM-
 109 based methods.

108
 109
 110
 111

- **Efficiency in Inference-Time Cost:** These scores are achieved using only SLMs with a
 smaller sample budget and a smaller token budget than previous state-of-the-art approaches,
 emphasizing the efficiency of our approach in terms of inference-time cost.

112
 113

2 RELATED WORK

114
 115 In this section, we provide a brief overview of recent advancements in automated formal theorem
 116 proving. Details of representative systems are provided in Appendix A.

117 **Tree-Search-based Formal Proving.** Tree-search methods construct Lean proofs tactic-by-tactic
 118 and navigate the proof space with explicit search, such as best-first search or Monte-Carlo tree
 119 search (MCTS) (Lample et al., 2022; Wang et al., 2023; Wu et al., 2024a; Zhou et al., 2024; Li et al.,
 120 2025; Xin et al., 2025a;b). This line began with stepwise tactic prediction guided by a goal state,
 121 and matured into systems that optimize the tactic policy, the search heuristic, and data curation for
 122 longer proofs.

123 **Whole-Proof Generation.** A complementary line to tree-search methods is whole-proof genera-
 124 tion (First et al., 2023), where a model emits an entire Lean script in one shot, often accompanied by
 125 a long chain-of-thought reasoning trace. This approach has progressed via expert-iteration pipelines
 126 that recycle verified proofs back into training (Polu et al., 2023; Wu et al., 2021; 2024a; Lin et al.,
 127 2025a; Dong & Ma, 2025; Lin et al., 2025b;c) and via reinforcement learning with formal verifier
 128 feedback (Kaliszyk et al., 2018; Xin et al., 2025a; Zhang et al., 2025; Wang et al., 2025; Ren et al.,
 129 2025; Gloeckle et al., 2024; Ji et al., 2025; Lin et al., 2025c).

130 **Formal Theorem Proving with Retrieval-Augmented Generation.** Another emerging direction is
 131 to combine LLM-based provers with retrieval-augmented generation (RAG), where external knowl-
 132 edge sources or proof libraries are queried at inference time to supplement the model’s reason-
 133 ing (Yang et al., 2023; Shen et al., 2025)

134 **Proof Refinement and Subgoal Decomposition.** Some work has explored proof refinement, where
 135 an initial proof attempt is improved based on feedback from the proof assistant (Thakur et al., 2024;
 136 Zhou et al., 2025; Chen et al., 2025; Lin et al., 2025c). Another line of work involves subgoal
 137 decomposition, where a complex theorem is broken down into simpler subgoals that are easier to
 138 prove (Dong et al., 2025; Wang et al., 2024a; Ren et al., 2025; Zhou et al., 2025), often guided by
 139 natural-language sketches (Jiang et al., 2023; Cao et al., 2025).

140 The subgoal decomposition approach shares certain similarities with ours, but our method adopts a
 141 more comprehensive strategy that subsumes it. In these works, the full sketch of the proof must be
 142 correctly envisioned upfront, which is often challenging. In contrast, our approach does not assume
 143 that the overall proof strategy is fully visible from the beginning. Rather than limiting decomposition
 144 to subgoals directly aligned with a pre-defined proof plan, we also consider auxiliary lemmas, such
 145 as special cases or potentially useful facts, to help develop a strategy in a bottom-up manner.

3 METHOD

146 The overall workflow is illustrated in Figure 2 and the corresponding pseudocode is shown in Al-
 147 gorithm 1. Given a formal math problem, our agent first attempts a direct proof, which is often
 148 sufficient for simpler problems. For more difficult problems that cannot be solved directly, it gen-
 149 erates auxiliary lemmas to uncover a viable proof strategy. These lemmas are then formalized and
 150 proved individually, and the resulting proven lemmas are used to synthesize a final proof of the orig-
 151 inal problem. Throughout this process, feedback from Lean is used to iteratively refine constructed
 152 proofs. We describe each stage below, highlighting how the informal LLM, formal prover model,
 153 and Lean coordinate to construct formal proofs.

3.1 FORMAL PROOF CONSTRUCTION GUIDED BY INFORMAL REASONING AND ITERATIVE FEEDBACK

154 The agent first attempts to directly prove the given problem or a generated lemma without decom-
 155 position. To leverage the stronger mathematical reasoning ability of the informal LLM compared to
 156 that of the formal prover model, we first generate an informal proof in natural language for the given

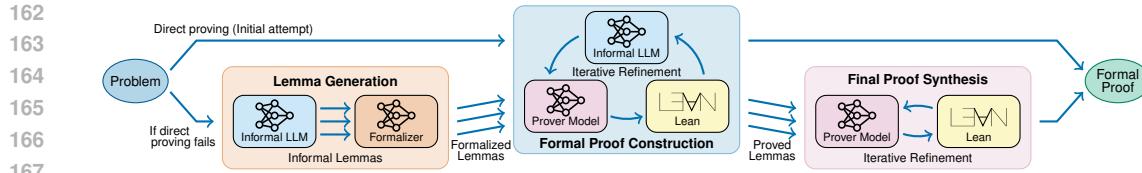


Figure 2: Overall workflow of Prover Agent. The agent coordinates informal reasoning, formal proving, and Lean verification. It first attempts direct proving; if unsuccessful, it generates auxiliary lemmas to guide the discovery of a viable proof strategy. These lemmas are then formally proved, and the successfully proved lemmas are subsequently used to synthesize the final proof.

problem or lemma using the informal LLM. The formal prover model then uses the informal proof as contextual guidance to generate a formal proof, which is subsequently verified by Lean. If the proof is successful, this step is complete. If the proof fails, these steps are repeated until a successful proof is found or the maximum number of attempts N_{init} is reached. This process helps establish a better initial outline for the subsequent iterative refinement process.

If the proof still fails, the agent enters an iterative refinement stage. The proof with the fewest Lean verification errors among the prior attempts is selected as the initial draft. This proof is then iteratively refined based on the feedback from Lean. In each iteration, the previous proof attempt, along with the error locations and corresponding error messages, is provided to the prover model, which revises and generates a corrected version of the proof. This process is repeated until the proof is successfully verified by Lean or the maximum number of attempts N_{refine} is reached.

This iterative refinement process leverages Lean’s verification to identify and correct mistakes. It serves as a form of self-correction through in-context learning, akin to how humans improve their understanding from feedback. This provides an efficient remedy to a key limitation of inference-time scaling with chain-of-thought, where simply increasing the number of reasoning steps does not guarantee better results due to the model’s limited ability of self-correction (Zeng et al., 2025; Song et al., 2025; Stechly et al., 2025; Huang et al., 2024).

It is accessible if a generated lemma cannot be proven. This mirrors how human mathematicians often approach problems: when the overall strategy is unclear at the beginning, they may explore several directions, some of which turn out to be unproductive and are eventually discarded in favor of more promising ones. Alternatively, to handle cases where the lemma is still too challenging to prove, the system may recursively introduce smaller auxiliary lemmas, up to a depth limit D .

3.2 LEMMA GENERATION VIA INFORMAL REASONING

When the direct proving approach fails to solve the problem, the agent generates several auxiliary lemmas. These are not limited to subgoals that can be directly inserted into a final proof; they may also include special cases or potentially useful facts derived from the assumptions that help in developing a proof strategy. This represents a key difference from prior work, which typically relies on decomposing the problem into subgoals based on a pre-defined proof sketch (Jiang et al., 2023; Wang et al., 2024a; Ren et al., 2025; Cao et al., 2025; Zhou et al., 2025). In such approaches, it is necessary to come up with the correct overall proof strategy beforehand, which is often a challenging task. Indeed, these methods often rely on larger, stronger models such as DeepSeek-V3 (DeepSeek-AI, 2024) and DeepSeek-R1 (DeepSeek-AI, 2025) to accurately predict the entire proof plan from the outset. In contrast, our approach does not assume that the proof strategy is visible from the outset. Instead, by generating auxiliary lemmas, the agent can gradually construct an effective proof strategy in a bottom-up manner, even when the full structure is not initially apparent.

For example, when trying to prove that $n^2 + an$ is even for a natural number n and an odd number a , it may be helpful to first consider special cases such as $a = 1$ or $a = 3$, i.e., $n^2 + n$ or $n^2 + 3n$. These special cases can help reveal patterns and guide the overall proof strategy for $n^2 + an$, even though expressions like $n^2 + n$ or $n^2 + 3n$ may not explicitly appear as steps within the final proof.

This approach mirrors how human mathematicians typically work. When the overall strategy is not clear at the beginning, they often explore special cases or consider what can be derived from the assumptions. Through such trial and error, they gradually discover the overall proof strategy.

216 The system first generates lemmas in natural language to leverage the stronger mathematical reasoning
 217 capabilities of the informal LLM. These lemmas are then converted into formal statements by a
 218 formalizer model, which formalizes only their assumptions and conclusions with no proof attempt.
 219 Lean is also used here to verify the syntactic correctness of the formalized statements, which are
 220 regenerated until they become valid. These formally stated lemmas are then proved using the proof
 221 construction process described in Section 3.1.

223 3.3 FINAL PROOF SYNTHESIS GUIDED BY VERIFIED LEMMAS AND ITERATIVE FEEDBACK

225 After attempting to prove each of these lemmas individually, the agent reconsiders the overall proof.
 226 With the verified lemmas as context, it attempts to construct a proof up to N_{init} times, followed by
 227 iterative refinement for up to N_{refine} attempts, as described in Section 3.1.

228 4 THEORETICAL ANALYSIS

231 We present theoretical analyses to justify the effectiveness of our approach described in Section 3.
 232 The use of lemmas serves two key purposes: (i) decomposing proof steps under a given strategy to
 233 make them more manageable, and (ii) helping discover proof strategies when the appropriate one is
 234 not initially clear (e.g., by testing special cases). Prior work has largely focused only on (i), often
 235 requiring larger models to directly devise an overall strategy (Wang et al., 2024a; Jiang et al., 2023;
 236 Ren et al., 2025; Cao et al., 2025; Zhou et al., 2025), whereas our approach leverages both (i) and (ii)
 237 to solve difficult problems more effectively. Sections 4.1 and 4.2 present brief results of theoretical
 238 analyses on lemma usage in cases (i) and (ii), respectively. See Appendix C for the details.

239 4.1 BENEFITS OF LEMMAS FOR STRUCTURED PROOF DECOMPOSITION

241 **Assumption 4.1.** For a certain class of theorems, it is necessary to satisfy m intermediate facts
 242 F_1, \dots, F_m , which correspond to subgoals that would typically appear as have statements in Lean.

243 **Assumption 4.2.** The probability p_i that the model correctly produces each F_i in a single attempt
 244 is independent across i within one global generation.

245 **Assumption 4.3.** Given a set of completed intermediate facts $\{F_i\}_{i \in S}$ with $S \subseteq [m]$ ¹, the probability
 246 of proving their composition F_S (e.g., simply concatenating them) is higher than the probability
 247 of proving F_S without being given those facts: $\mathbb{P}(F_S \mid \{F_i\}_{i \in S}) > \mathbb{P}(F_S)$.

248 Assuming $p = p_1 = \dots = p_m$ for simplicity, the following theorems hold. Rigorous versions
 249 without this simplification and without asymptotic notation are provided in Appendix C.1.

251 **Theorem 4.4** (Required Number of Trials). *Let N_{dir} denote the number of trials required to directly
 252 prove a problem T with probability at least $1 - \delta$. Let N_{lem} denote the total number of trials required
 253 to complete the proof of T with probability at least $1 - \delta$, when lemmas L_1, \dots, L_n are introduced
 254 with an allowed failure probability δ_{lem} . Suppose each lemma L_i contains a subset of the essential
 255 intermediate facts $\{F_i\}_{i \in S_i}$ with $S_i \subseteq [m]$. Then the following holds:*

$$256 N_{\text{dir}} = \Theta(p^{-m}), \quad \mathbb{E}[N_{\text{lem}}] = \tilde{\Theta}(p^{-s}),$$

257 where $s := \max\{\max_i |S_i|, |R_0|\} \leq m$, $R_0 := [m] \setminus \bigcup_{i=1}^n S_i$, and $\tilde{\Theta}$ indicates asymptotic order
 258 ignoring higher-order terms in δ_{lem} , which vanish when δ_{lem} is sufficiently small.

259 **Theorem 4.5** (Threshold Condition for Lemma Efficiency). *There exists a threshold $\tau \in [0, 1]$ such
 260 that if $p \leq \tau$, then $\mathbb{E}[N_{\text{lem}}] \leq N_{\text{dir}}$ holds for any $\delta, \delta_{\text{lem}} \in (0, 1)$.*

261 **Theorem 4.6** (Optimal Partition of Lemma Coverage). *Under the fixed lemma coverage $U :=$
 262 $\bigcup_{i=1}^n S_i \subseteq [m]$, $\mathbb{E}[N_{\text{lem}}]$ is minimized when $|S_i| = \lceil |U|/n \rceil$ or $\lfloor |U|/n \rfloor$ for all $i \in [n]$.*

263 The proofs are provided in Appendix C.1. Theorem 4.4 shows that lemma-based decomposition
 264 yields an exponential improvement in the order of required trials, while Theorem 4.5 indicates that
 265 for small p (i.e., difficult problems), lemma usage reduces the required number of trials. This jus-
 266 tifies our approach of generating lemmas for difficult problems while solving easy ones directly.
 267 Furthermore, Theorem 4.6 suggests that the optimal lemmas are those that divide the problem into
 268 subproblems of approximately equal difficulty.

269 ¹[m] denotes the set $\{1, 2, \dots, m\}$.

270 4.2 BENEFITS OF LEMMAS FOR DISCOVERING PROOF STRATEGIES (E.G., SPECIAL CASES)
271

272 Let \mathcal{S} be the set of possible proof strategies (e.g., induction, bounding with monotonicity, or case
273 analysis with known results). Let π_0 denote the prior distribution over strategies that the model
274 possesses, from which a strategy is chosen in the absence of any additional information. Our agent
275 conducts experiments with lemmas L_1, \dots, L_n and verifies them in Lean, thereby obtaining obser-
276 vations Y_1, \dots, Y_n . By incorporating these observations into the context, the distribution is updated
277 to the posterior $\pi_n(\cdot) := \pi(\cdot \mid Y_{1:n})$, where $Y_{1:n} := \{Y_1, \dots, Y_n\}$, aiming to increase the probability
278 of selecting the correct proof strategy.

279 Let $p(z) \in [0, 1]$ denote the model’s success probability under a given strategy $z \in \mathcal{S}$, and define
280 $r := \inf_z p(z)$. As shown in Section 4.1, this quantity can be increased by using decomposition-type
281 lemmas. Define the entropy of the prior distribution as $H_0 := H(Z) = -\sum_{z \in \mathcal{S}} \pi_0(z) \log \pi_0(z)$.

282 **Theorem 4.7** (Success Probability Improvement by Lemmas). *The success probability of performing
283 one trial of final proving by sampling a strategy from the posterior distribution π_n is bounded as
284 follows:*

$$285 \mathbb{E}[\mathbb{P}(\text{succ@1})] \geq r \exp(-H_0 + I(Z; Y_{1:n})).$$

286 The proof is provided in Appendix C.2. This shows that the success probability improves expo-
287 nentially in the mutual information contributed by lemmas, $I(Z; Y_{1:n})$. In particular, it exceeds the
288 no-lemma case, where $I(Z; Y_{1:n}) = 0$.

289 Furthermore, this result implies that not only lemmas but any information in the context that shares
290 mutual information with the final correct proof can similarly improve the success probability, thereby
291 justifying our use of natural language proofs and Lean feedback.

293 5 EXPERIMENTS
294295 5.1 EXPERIMENTAL SETUP
296

297 We evaluate our approach on both the MiniF2F benchmark (Zheng et al., 2022) and Putnam-
298 Bench (Tsoukalas et al., 2024b), two widely used datasets for assessing formal theorem-proving sys-
299 tems. We use DeepSeek-R1-0528-Qwen3-8B (DeepSeek-AI, 2025) for the informal reasoning
300 LLM and DeepSeek-Prover-V2-7B (Ren et al., 2025) and Goedel-Prover-V2-8B (Lin
301 et al., 2025c) for the prover model. We set $N_{\text{init}} = N_{\text{refine}} = 50$. Thus, the sample budget at
302 the initial direct proving stage is 50 at the first iteration, and 100 in total when including itera-
303 tive refinement. For lemmas, we use $N_{\text{init}} = N_{\text{refine}} = 10$ for each of the three lemmas. In the
304 final synthesis stage, $N_{\text{init}} = N_{\text{refine}} = 50$ is used again, resulting in a total sample budget of
305 $50 + 50 + (10 + 10) \times 3 + 50 + 50 = 260$. The maximum decomposition depth D is set to 1. All
306 prompts used in our experiments are provided in Appendix F. All runs are performed on NVIDIA
307 40GB A100 GPUs with vLLM (Kwon et al., 2023). See Appendix D for further details.

308 There are several bugs that may result in invalid Lean proofs being incorrectly accepted, such as
309 the user-interference bug related to the `apply?` tactic discussed in Ren et al. (2025), and a bug in
310 REPL². To avoid these issues and prevent invalid proofs from being mistakenly judged as correct, we
311 check proofs with `lake build` instead of REPL and additionally verified that the `apply?` tactic
312 is not used. Also, to avoid this bug and obtain reliable baseline results, we re-run the experiments
313 for Goedel-Prover-V2-8B. We use the official prompts provided on GitHub³ and Hugging Face⁶,
314 while keeping all other experimental settings strictly identical to those used in our method, thereby
315 ensuring a fair comparison. For DeepSeek-Prover-V2, we relied on the results reported in (Ren
316 et al., 2025), in which this bug has been fixed. See Appendix D for further details.

317 5.2 MAIN RESULT: COMPARISON WITH THE PREVIOUS STATE-OF-THE-ART
318

319 The results are shown in Table 1, Table 2, and Figure 1. On the MiniF2F benchmark, our agent
320 achieves an 88.1% success rate, establishing a new state-of-the-art among methods using small lan-
321 guage models (SLMs). Note that our agent achieves this result with a sample budget of only 260,

322 ²<https://github.com/leanprover-community/repl/issues/44>

323 ³<https://github.com/Goedel-LM/Goedel-Prover-V2>

324
 325
 326
 327
 328
 329 Table 1: Comparison of formal theorem-proving performance on miniF2F-test. The results are re-
 330 ported as the percentage of theorems proved correctly. For Prover Agent, sample budget includes all
 331 proof attempts across the full pipeline, including initial direct proving, iterative refinement, lemma
 332 proving, and final proof synthesis. The best results within each model scale are highlighted in **bold**.
 333
 334
 335
 336
 337

Prover System	Method	Model Size	Sample Budget	Success Rate
<i>Large Language Models</i>				
<i>Large Language Models</i>				
DSP+ (Cao et al., 2025) w/ QwQ, DeepSeek-V3, and BFS-Prover	Informal + Tree search	671B	1	52.5%
			128	74.2%
w/ DeepSeek-R1, DeepSeek-V3, and BFS-Prover			1024	79.5%
			1024	80.7%
DeepSeek-Prover-V2 (Ren et al., 2025)	Whole-proof	671B	1	61.9%
			1024	86.6%
			8192	88.9%
Delta-Prover (Zhou et al., 2025) w/ Gemini 2.5 Pro	Agent	unknown	16384	95.9%
Seed-Prover (Chen et al., 2025)	Whole-proof	unknown	unknown	99.6%
<i>Medium Language Models</i>				
<i>Medium Language Models</i>				
Kimina-Prover-Preview (Wang et al., 2025)	Whole-proof	72B	1	52.9%
			1024	77.9%
			8192	80.7%
Goedel-Prover-V2 (Lin et al., 2025c)	Whole-proof	32B	32	88.1%
			1024	91.8%
			8192	92.2%
<i>Small Language Models</i>				
<i>Small Language Models</i>				
DeepSeek-Prover-V1.5-RL + RMaxTS (Xin et al., 2025a)	Tree search	7B	32 × 16 × 400	63.5%
InternLM2.5-StepProver-BF + CG (Wu et al., 2024a)	Tree search	7B	256 × 32 × 600	65.9%
HunyuanProver v16 + BFS + DC (Li et al., 2025)	Tree search	7B	600 × 8 × 400	68.4%
BFS-Prover (Xin et al., 2025b)	Tree search	7B	2048 × 2 × 600	70.8%
Leanabell-Prover-GD-RL (Zhang et al., 2025)	Whole-proof	7B	128	61.1%
Goedel-Prover-SFT (Lin et al., 2025b)	Whole-proof	7B	25600	64.7%
STP (Dong & Ma, 2025)	Whole-proof	7B	25600	67.6%
Kimina-Prover-Preview-Distill (Wang et al., 2025)	Whole-proof	7B	1	52.5%
			32	63.1%
			1024	70.8%
DeepSeek-Prover-V2 (Ren et al., 2025)	Whole-proof	7B	1	58.6%
			32	75.6%
			1024	79.9%
			8192	82.0%
Leanabell-Prover-V2-KM (Ji et al., 2025)	Whole-proof	7B	32	68.4%
Leanabell-Prover-V2-DS (Ji et al., 2025)	Whole-proof	7B	128	70.4%
Goedel-Prover-V2 (Lin et al., 2025c)	Whole-proof	8B	1	60.8%
			64	83.3%
			256	85.2%
			512	85.7%
w/ DeepSeek-Prover-V2 (Direct proving w/o iterative refinement) (Direct proving w/o iterative refinement) (Direct proving w/ iterative refinement) (Final proof synthesis w/ lemma)			1	61.5%
			50	79.9%
			100	82.0%
			260	82.8%
Prover Agent (Ours) w/ Goedel-Prover-V2 (Direct proving w/o iterative refinement) (Direct proving w/o iterative refinement) (Direct proving w/ iterative refinement) (Final proof synthesis w/ lemma)	Agent	8B	1	64.3%
			50	84.4%
			100	85.7%
			260	86.5%
w/ Ensemble of Goedel-Prover-V2 and DeepSeek-Prover-V2 (Direct proving w/o iterative refinement) (Direct proving w/o iterative refinement) (Direct proving w/ iterative refinement) (Final proof synthesis w/ lemma)			1	64.3%
			50	85.7%
			100	86.9%
			260	88.1%

369
 370
 371 far smaller than that of prior work, highlighting its efficiency in inference-time cost. Moreover, even
 372 when evaluated in terms of the total token budget consumed across all LLM calls, our approach
 373 achieves higher success rates with a smaller token budget than the baselines, demonstrating its over-
 374 all efficiency (see Appendix D.6 for details). Furthermore, on the more challenging PutnamBench,
 375 Prover Agent solves 25 problems with a sample budget of only 110. This surpasses the baseline
 376 score despite using fewer samples, establishing a new state-of-the-art among methods based on
 377 SLMs. The consistent improvements observed across both MiniF2F and PutnamBench underscore
 the robustness and generality of our approach.

378
 379 Table 2: Comparison of formal theorem-proving performance on PutnamBench. The results are
 380 reported as the number of theorems proved correctly. For Prover Agent, sample budget includes all
 381 proof attempts across the full pipeline, including initial direct proving, iterative refinement, lemma
 382 proving, and final proof synthesis. The best results within each model scale are highlighted in **bold**.
 383

Prover System	Method	Model Size	Sample Budget	# Solved
<i>Large Language Models</i>				
DSP+ (Cao et al., 2025)	Informal + Tree search	671B	1024	25/644
DeepSeek-Prover-V2 (Ren et al., 2025)	Whole-proof	671B	32 128 1024	22/658 33/658 47/658
<i>Medium Language Models</i>				
Goedel-Prover-V2 (Lin et al., 2025c)	Whole-proof	32B	32 184	57/644 86/644
<i>Small Language Models</i>				
InternLM2.5-StepProver-BF + CG (Wu et al., 2024a) STP (Dong & Ma, 2025)	Tree search Whole-proof	7B 7B	2 × 32 × 600 3200	6/640 8/644
Goedel-Prover-SFT (Lin et al., 2025b)	Whole-proof	7B	32 512	6/644 7/644
Kimina-Prover-Preview-Distill (Wang et al., 2025)	Whole-proof	7B	192	10/644
DeepSeek-Prover-V2 (Ren et al., 2025)	Whole-proof	7B	32 128 1024	9/658 10/658 11/658
Goedel-Prover-V2 (Lin et al., 2025c)	Whole-proof	8B	32 128	18/659 22/659
Prover Agent (Ours) w/ Goedel-Prover-V2	(Direct proving w/ iterative refinement) (Final proof synthesis w/ lemma)	Agent	40 110	20/659 25/659

405 5.3 MODULAR AND SCALABLE DESIGN

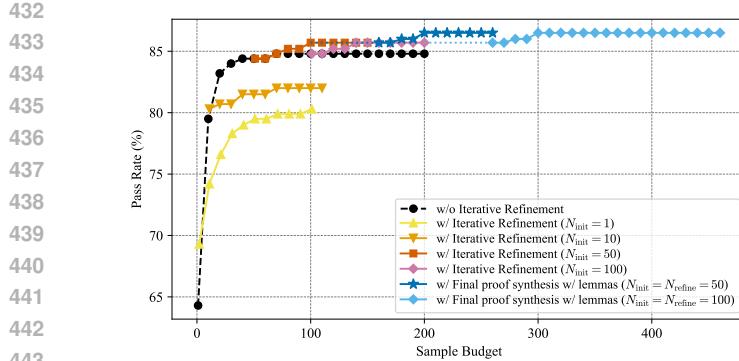
406
 407 To demonstrate the robustness of our approach, we conduct experiments across several models,
 408 namely DeepSeek-Prover-V2 and Goedel-Prover-V2. In both settings, our approach achieves higher
 409 success rates with a smaller sample budget than the vanilla versions of these models, as shown in
 410 Table 1. Furthermore, our approach can also ensemble these models. In experiments where the
 411 sample budget is split evenly between them, our agent achieves an even higher success rate, where
 412 the models complement each other on problems that one alone cannot solve. Unlike monolithic
 413 approaches that train a single large model end-to-end, our method takes an orthogonal approach by
 414 combining an existing LLM and a prover model without any training. This modular design provides
 415 a practical benefit, allowing the system to immediately take advantage of improvements in LLMs
 416 and prover models by simply replacing components and to scale easily with future advancements.

417 5.4 EFFECTIVENESS OF INFORMAL, FORMAL, AND LEAN COORDINATION

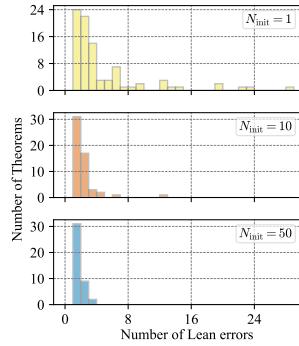
418
 419 Table 1 shows that in both model settings, our approach outperforms the corresponding vanilla
 420 baselines even before the iterative refinement, highlighting the benefit of collaboration with the
 421 informal LLM. Moreover, the scores increase even further after iterative refinement.

423 5.5 ABLATION STUDIES: ANALYZING THE CONTRIBUTION OF EACH STAGE

424
 425 We conduct ablation studies to illustrate the contribution of each stage of our agent. Results for
 426 different N_{init} and N_{refine} are shown in Figure 3a. When N_{init} is set to 1 or 10, the success rate
 427 remains significantly lower than that without iterative refinement, even after $N_{\text{refine}} = 100$ refine-
 428 ment steps. This highlights the importance of the quality of the initial draft used to start refinement:
 429 if the initial proof is poor, subsequent refinement becomes significantly more difficult (The case
 430 study in Appendix E.2 shows that refinement depends on the original Lean code and addresses its
 431 errors). Comparing $N_{\text{init}} = 1, 10, 50$ under the same sample budget shows a clear improvement in
 432 performance in this order, indicating the effectiveness of our approach of selecting the proof with



(a) Results for different N_{init} and N_{refine} . The dotted lines indicate that the corresponding sample budget are used in the proof of lemmas.



(b) Histogram of Lean error counts after N_{init} .

Figure 3: Ablation study results on N_{init} and N_{refine} . These results highlight the importance of initial draft selection and indicate that iterative refinement and lemma-based proving helps overcome saturation from the model’s inherent limitations.

the fewest Lean errors. As shown in Figure 3b, the histograms of the minimum number of errors after $N_{\text{init}} = 1, 10, 50$ confirm this trend: the error count decreases substantially, and for $N_{\text{init}} = 50$ most problems have only one or two errors. Although the number of Lean errors may not perfectly measure proof quality, since a single error can still correspond to a mathematically challenging gap, it nevertheless exhibits a strong correlation and serves as a useful proxy for evaluation.

As shown in Figure 3a, the runs without iterative refinement saturate around a sample budget of 80. In contrast, when iterative refinement is applied after $N_{\text{init}} = 50$ or 100, this saturation is overcome and the success rate improves, outperforming the setting that simply continues generation without refinement. This demonstrates the effectiveness of the iterative refinement: whereas repeated generation alone eventually saturates due to the inherent ability limits of the model, incorporating external feedback through in-context learning enables the model to improve and overcome this limitation. Also, $N_{\text{init}} = 50$ and 100 yield almost identical results in the final performance. Since the model had already saturated in this regime, increasing N_{init} did not improve the quality of the selected initial drafts. Furthermore, Figure 3a shows that final synthesis with lemmas improves the score even after iterative refinement has saturated, demonstrating the effectiveness of our lemma-based approach. This indicates that the model’s capability is further enhanced by incorporating information beyond mere error feedback.

5.6 CASE STUDY: SUCCESS WITH LEMMA-GUIDED PROOFS AND ITERATIVE REFINEMENT

We next present a case study to demonstrate that our approach with auxiliary lemmas is indeed effective in practice. The detailed discussion and the outputs for this problem, such as the generated lemmas, final formal proof, and the associated reasoning process, are provided in Appendix E.1. We analyze the output and reasoning process for the problem where the direct proof attempt failed but the use of auxiliary lemmas led to a successful proof. The case study illustrating an example that succeeds through iterative refinement is described in detail in Appendix E.2, where it demonstrates how providing feedback about Lean’s limitations helps guide the model toward constructing an effective proof.

In this case, our agent generates a lemma corresponding to the special case of substituting $n = 3$ into the given problem, as well as additional lemmas that may be potentially relevant for solving the problem. As observed in the chain-of-thought process when this lemma is used (see Appendix E.1.5), the agent immediately considers the $n = 3$ case and then quickly comes up with mathematical induction as the proof strategy. This allows it to quickly transition to filling in the details under a clear proof plan and ultimately complete the proof. Moreover, tactics and proof techniques considered in the auxiliary lemmas reappear in the reasoning process and final proof: even when a lemma itself is not directly used, the techniques explored during lemma generation provide valuable hints for the overall proof construction.

Next, for comparison, we examine the reasoning process without using lemmas, focusing on the trajectory with the fewest final errors (see Appendix E.1.6). Compared to the successful case with

486 Table 3: Comparison of formal theorem-proving performance by problem category on MiniF2F-test.
 487 The results are reported as the percentage of theorems proved. The best results in each model setting
 488 for each of the three categories, demarcated by double lines, are highlighted in **bold**.

	Model Size	Sample Budget	Olympiad				MATH				Custom				
			IMO	AIME	AMC	Sum	Algebra	Number Theory	Sum	Algebra	Number Theory	Induction	Sum		
Number of Problems			20	15	45	80	70	60	130	18	8	8	34		
DeepSeek-Prover-V2 (Ren et al., 2025)	671B	8192	50.0	93.3	77.8	73.8	100.0	96.7	98.5	83.3	87.5	100.0	88.2		
Prover Agent (Ours)			1	40.0	53.3	62.2	55.0	71.4	60.0	66.2	55.6	75.0	50.0	58.8	
w/ DeepSeek-Prover-V2			8B	50	70.0	80.0	82.2	78.8	80.0	88.3	83.8	66.7	75.0	62.5	67.6
(Direct proving w/o iterative refinement)				100	70.0	80.0	86.7	81.3	84.3	88.3	86.2	66.7	75.0	62.5	67.6
(Direct proving w/ iterative refinement)				260	70.0	80.0	88.9	82.5	84.3	88.3	86.2	66.7	75.0	70.6	
(Final proof synthesis w/ lemma)															
Goedel-Prover-V2 (Lin et al., 2025c)			1	50.0	60.0	53.3	53.8	71.4	63.3	67.7	50.0	62.5	50.0	52.9	
			8B	64	80.0	80.0	88.9	85.0	84.3	91.7	87.7	77.8	75.0	87.5	79.4
				256	80.0	80.0	88.9	85.0	84.3	91.7	87.7	77.8	75.0	87.5	79.4
				512	80.0	80.0	88.9	85.0	84.3	91.7	87.7	77.8	75.0	87.5	79.4
Prover Agent (Ours)			1	50.0	73.3	57.8	58.8	68.6	70.0	69.2	55.6	62.5	62.5	58.8	
w/ Goedel-Prover-V2			8B	50	80.0	80.0	86.7	83.8	84.3	90.0	86.9	77.8	75.0	75.0	76.5
(Direct proving w/o iterative refinement)				100	80.0	80.0	88.9	85.0	87.1	90.0	88.5	77.8	75.0	75.0	76.5
(Direct proving w/ iterative refinement)				260	80.0	80.0	88.9	85.0	88.6	90.0	89.2	77.8	75.0	87.5	79.4
Prover Agent (Ours)			1	50.0	73.3	57.8	58.8	68.6	70.0	69.2	55.6	62.5	62.5	58.8	
w/ Ensemble			8B	50	80.0	80.0	88.9	85.0	87.1	90.0	88.5	77.8	75.0	75.0	76.5
(Direct proving w/o iterative refinement)				100	80.0	80.0	91.1	86.3	90.0	90.0	90.0	77.8	75.0	75.0	76.5
(Direct proving w/ iterative refinement)				260	80.0	80.0	93.3	87.5	91.4	90.0	90.8	77.8	75.0	87.5	79.4
(Final proof synthesis w/ lemma)															

503
 504 lemmas, the proof strategy here is far less clear, with the model wandering without a coherent plan.
 505 As a result, even when it eventually reaches the idea of using mathematical induction, it fails to
 506 elaborate on the details, and the proof does not succeed. This comparison highlights the effectiveness
 507 of our auxiliary-lemma approach, which goes beyond the simple decomposition of previous work.
 508

509 5.7 PERFORMANCE ON OLYMPIAD-LEVEL PROBLEMS

510
 511 Table 3 shows the results for each category on the MiniF2F-test dataset. These results demonstrate
 512 that our approach with DeepSeek-Prover-V2 setting performs particularly well on Olympiad-level
 513 problems, even surpassing DeepSeek-Prover-V2 (Ren et al., 2025), which uses a significantly larger
 514 671B model and a much higher sample budget of 8192. Given that our direct proving method with-
 515 out iterative refinement and with a sample budget of only 100 already surpasses DeepSeek-Prover-
 516 V2, this suggests that coordination with natural language-based informal reasoning may be the key.
 517 Olympiad-level problems require a high degree of mathematical reasoning, and the strong reasoning
 518 abilities of the informal LLM likely played a crucial role in solving them effectively. On the other
 519 hand, our agent does not outperform DeepSeek-Prover-V2 in the MATH and Custom categories.
 520 The consistent gap in these categories suggests that model size and sample budget may play a more
 521 significant role here. Since DeepSeek-Prover-V2 also possesses a certain level of mathematical
 522 reasoning ability, it can handle these relatively mathematically easier problems on its own. In contrast,
 523 with the Goedel-Prover-V2 setting, no substantial differences are observed across categories. This
 524 is likely because Goedel-Prover-V2 already possesses a certain level of the required mathematical
 525 capability for all these categories, and thus category-specific variation does not emerge as clearly.

526 5.8 BROADER APPLICABILITY AND FUTURE POTENTIAL

527 Nothing in our pipeline is specific to mathematics competition problems. The same approach could
 528 be applied to formal proofs in other domains, such as learning theory or physics, as long as the LLM
 529 has relevant knowledge or is provided with an appropriate knowledge base. This offers the potential
 530 for AI-driven construction of mathematical theories without hallucinations or logical errors.
 531

532 6 CONCLUSION

533
 534 We introduced Prover Agent, a modular framework that coordinates an informal reasoning LLM,
 535 a formal prover model, and Lean verification. By generating auxiliary lemmas and leveraging
 536 feedback-driven refinement, our method achieved state-of-the-art performance among methods using
 537 SLMs on both MiniF2F PutnamBench. Future work includes developing mechanisms to generate
 538 more effective lemmas tailored to different types of problems, and extending our framework to
 539 domains beyond mathematics that require formal verification, such as software verification.

540 REFERENCES
541

542 Thomas Anthony, Zheng Tian, and David Barber. Thinking fast and slow with deep learning and
543 tree search. In *Advances in Neural Information Processing Systems*, volume 30, 2017.

544 Zhangir Azerbayev, Bartosz Piotrowski, Hailey Schoelkopf, Edward W. Ayers, Dragomir Radev,
545 and Jeremy Avigad. ProofNet: Autoformalizing and formally proving undergraduate-level math-
546 ematics. *arXiv preprint arXiv:2302.12433*, 2023.

547 Bruno Barras, Samuel Boutin, Cristina Cornes, Judicaël Courant, Yann Coscoy, David Delahaye,
548 Daniel de Rauglaudre, Jean-Christophe Filliâtre, Eduardo Giménez, Hugo Herbelin, et al. The
549 Coq proof assistant reference manual. *INRIA, version*, 6(11):17–21, 1999.

550 Chenrui Cao, Liangcheng Song, Zenan Li, Xinyi Le, Xian Zhang, Hui Xue, and Fan Yang. Re-
551 living DSP for advanced theorem proving in the era of reasoning models. *arXiv preprint*
552 *arXiv:2506.11487*, 2025.

553 Luoxin Chen, Jinming Gu, Liankai Huang, Wenhao Huang, Zhicheng Jiang, Allan Jie, Xiaoran Jin,
554 Xing Jin, Chenggang Li, Kaijing Ma, Cheng Ren, Jiawei Shen, Wenlei Shi, Tong Sun, He Sun,
555 Jiahui Wang, Siran Wang, Zhihong Wang, Chenrui Wei, Shufa Wei, Yonghui Wu, Yuchen Wu,
556 Yihang Xia, Huajian Xin, Fan Yang, Huaiyuan Ying, Hongyi Yuan, Zheng Yuan, Tianyang Zhan,
557 Chi Zhang, Yue Zhang, Ge Zhang, Tianyun Zhao, Jianqiu Zhao, Yichi Zhou, and Thomas Hanwen
558 Zhu. Seed-Prover: Deep and broad reasoning for automated theorem proving. *arXiv preprint*
559 *arXiv:2507.23726*, 2025.

560 DeepSeek-AI. DeepSeek-V3 technical report. *arXiv preprint arXiv:2412.19437*, 2024.

561 DeepSeek-AI. DeepSeek-R1: Incentivizing reasoning capability in llms via reinforcement learning.
562 *arXiv preprint arXiv:2501.12948*, 2025.

563 Kefan Dong and Tengyu Ma. STP: Self-play llm theorem provers with iterative conjecturing and
564 proving. *arXiv preprint arXiv:2502.00212*, 2025.

565 Kefan Dong, Arvind Mahankali, and Tengyu Ma. Formal theorem proving by rewarding llms to
566 decompose proofs hierarchically. *arXiv preprint arXiv:2411.01829*, 2025.

567 Emily First, Markus N. Rabe, Talia Ringer, and Yuriy Brun. Baldur: Whole-proof generation and
568 repair with large language models. ESEC/FSE 2023, pp. 1229–1241. Association for Computing
569 Machinery, 2023. ISBN 9798400703270. doi: 10.1145/3611643.3616243.

570 Fabian Gloeckle, Jannis Limberg, Gabriel Synnaeve, and Amaury Hayat. ABEL: Sample efficient
571 online reinforcement learning for neural theorem proving. In *The 4th Workshop on Mathematical*
572 *Reasoning and AI at NeurIPS’24*, 2024.

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

576 Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn
577 Song, and Jacob Steinhardt. Measuring mathematical problem solving with the MATH dataset.
578 In *Thirty-fifth Conference on Neural Information Processing Systems Datasets and Benchmarks*
579 *Track (Round 2)*, 2021.

580 Jie Huang, Xinyun Chen, Swaroop Mishra, Huaixiu Steven Zheng, Adams Wei Yu, Xinying Song,
581 and Denny Zhou. Large language models cannot self-correct reasoning yet. In *The Twelfth*
582 *International Conference on Learning Representations*, 2024.

583 Lei Huang, Weijiang Yu, Weitao Ma, Weihong Zhong, Zhangyin Feng, Haotian Wang, Qianglong
584 Chen, Weihua Peng, Xiaocheng Feng, Bing Qin, and Ting Liu. A survey on hallucination in large
585 language models: Principles, taxonomy, challenges, and open questions. *ACM Transactions on*
586 *Information Systems*, 43(2), 2025. ISSN 1046-8188. doi: 10.1145/3703155.

587 Geoffrey Irving, Christian Szegedy, Alexander A Alemi, Niklas Een, Francois Chollet, and Josef Ur-
588 ban. DeepMath - deep sequence models for premise selection. In *Advances in Neural Information*
589 *Processing Systems*, volume 29, 2016.

594 Xingguang Ji, Yahui Liu, Qi Wang, Jingyuan Zhang, Yang Yue, Rui Shi, Chenxi Sun, Fuzheng
 595 Zhang, Guorui Zhou, and Kun Gai. Leanabell-Prover-V2: Verifier-integrated reasoning for formal
 596 theorem proving via reinforcement learning. *arXiv preprint arXiv:2507.08649*, 2025.

597

598 Ziwei Ji, Nayeon Lee, Rita Frieske, Tiezheng Yu, Dan Su, Yan Xu, Etsuko Ishii, Ye Jin Bang,
 599 Andrea Madotto, and Pascale Fung. Survey of hallucination in natural language generation. *ACM
 600 Computing Surveys*, 55(12), 2023. ISSN 0360-0300. doi: 10.1145/3571730.

601 Albert Q. Jiang, Wenda Li, Jesse Michael Han, and Yuhuai Wu. LISA: Language models of isabelle
 602 proofs. In *6th Conference on Artificial Intelligence and Theorem Proving*, 2021.

603

604 Albert Q. Jiang, Sean Welleck, Jin Peng Zhou, Wenda Li, Jiacheng Liu, Mateja Jamnik, Timothée
 605 Lacroix, Yuhuai Wu, and Guillaume Lample. Draft, sketch, and prove: Guiding formal theorem
 606 provers with informal proofs. In *The Eleventh International Conference on Learning Representations*, 2023.

607

608 Cezary Kaliszyk, Josef Urban, Henryk Michalewski, and Miroslav Olšák. Reinforcement learning
 609 of theorem proving. In *Advances in Neural Information Processing Systems*, volume 31, 2018.

610

611 Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph
 612 Gonzalez, Hao Zhang, and Ion Stoica. Efficient memory management for large language model
 613 serving with pagedattention. In *Proceedings of the 29th Symposium on Operating Systems Principles*, pp. 611–626. Association for Computing Machinery, 2023. ISBN 9798400702297. doi:
 614 10.1145/3600006.3613165.

615

616 Guillaume Lample, Timothee Lacroix, Marie-Anne Lachaux, Aurelien Rodriguez, Amaury Hayat,
 617 Thibaut Lavigil, Gabriel Ebner, and Xavier Martinet. Hypertree proof search for neural theorem
 618 proving. In *Advances in Neural Information Processing Systems*, volume 35, pp. 26337–26349,
 619 2022.

620

621 Aitor Lewkowycz, Anders Andreassen, David Dohan, Ethan Dyer, Henryk Michalewski, Vinay Ra-
 622 masesh, Ambrose Sloane, Cem Anil, Imanol Schlag, Theo Gutman-Solo, Yuhuai Wu, Behnam
 623 Neyshabur, Guy Gur-Ari, and Vedant Misra. Solving quantitative reasoning problems with lan-
 624 guage models. In *Advances in Neural Information Processing Systems*, volume 35, pp. 3843–
 625 3857, 2022.

626

627 Yang Li, Dong Du, Linfeng Song, Chen Li, Weikang Wang, Tao Yang, and Haitao Mi. Hunyuan-
 628 Prover: A scalable data synthesis framework and guided tree search for automated theorem prov-
 629 ing. *arXiv preprint arXiv:2412.20735*, 2025.

630

631 Haohan Lin, Zhiqing Sun, Sean Welleck, and Yiming Yang. Lean-STaR: Learning to interleave
 632 thinking and proving. In *The Thirteenth International Conference on Learning Representations*,
 633 2025a.

634

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

638

639 Yong Lin, Shange Tang, Bohan Lyu, Ziran Yang, Jui-Hui Chung, Haoyu Zhao, Lai Jiang, Yihan
 640 Geng, Jiawei Ge, Jingruo Sun, Jiayun Wu, Jiri Gesi, Ximing Lu, David Acuna, Kaiyu Yang,
 641 Hongzhou Lin, Yejin Choi, Danqi Chen, Sanjeev Arora, and Chi Jin. Goedel-Prover-V2: Scal-
 642 ing formal theorem proving with scaffolded data synthesis and self-correction. *arXiv preprint
 643 arXiv:2508.03613*, 2025c.

644

645 Pan Lu, Liang Qiu, Wenhao Yu, Sean Welleck, and Kai-Wei Chang. A survey of deep learning
 646 for mathematical reasoning. In *Proceedings of the 61st Annual Meeting of the Association for
 647 Computational Linguistics (Volume 1: Long Papers)*, pp. 14605–14631. Association for Compu-
 648 tational Linguistics, 2023. doi: 10.18653/v1/2023.acl-long.817.

649

650 The mathlib Community. The lean mathematical library. In *Proceedings of the 9th ACM SIG-
 651 PLAN International Conference on Certified Programs and Proofs*, pp. 367–381. Association for
 652 Computing Machinery, 2020. ISBN 9781450370974. doi: 10.1145/3372885.3373824.

648 Norman D. Megill and David A. Wheeler. *Metamath: A Computer Language for Pure Mathematics*,
 649 2019. URL <http://us.metamath.org/downloads/metamath.pdf>.

650

651 Leonardo de Moura and Sebastian Ullrich. The lean 4 theorem prover and programming lan-
 652 guage. In *Automated Deduction—CADE 28: 28th International Conference on Automated*
 653 *Deduction, Virtual Event, LNCS 12699*, pp. 625–635. Springer-Verlag, 2021. doi: 10.1007/
 654 978-3-030-79876-5_37.

655

656 A. Newell and H. Simon. The logic theory machine—a complex information processing system. *IRE*
 657 *Transactions on Information Theory*, 2(3):61–79, 1956. doi: 10.1109/TIT.1956.1056797.

658

659 OpenAI. OpenAI o1 system card. *arXiv preprint arXiv:2412.16720*, 2024.

660

661 OpenAI. OpenAI o3-mini, 2025. URL <https://openai.com/index/openai-o3-mini/>.

662

663 Lawrence C. Paulson. *Isabelle a Generic Theorem Prover*. Springer Verlag, 1994.

664

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

667

668 Stanislas Polu, Jesse Michael Han, Kunhao Zheng, Mantas Baksys, Igor Babuschkin, and Ilya
 669 Sutskever. Formal mathematics statement curriculum learning. In *The Eleventh International*
Conference on Learning Representations, 2023.

670

671 Rafael Rafailov, Archit Sharma, Eric Mitchell, Christopher D Manning, Stefano Ermon, and Chelsea
 672 Finn. Direct preference optimization: Your language model is secretly a reward model. In *Ad-*
673 vances in Neural Information Processing Systems, volume 36, pp. 53728–53741, 2023.

674

675 Z. Z. Ren, Zhihong Shao, Junxiao Song, Huajian Xin, Haocheng Wang, Wanja Zhao, Liyue Zhang,
 676 Zhe Fu, Qihao Zhu, Dejian Yang, Z. F. Wu, Zhibin Gou, Shirong Ma, Hongxuan Tang, Yux-
 677 uan Liu, Wenjun Gao, Daya Guo, and Chong Ruan. DeepSeek-Prover-V2: Advancing formal
 678 mathematical reasoning via reinforcement learning for subgoal decomposition. *arXiv preprint*
arXiv:2504.21801, 2025.

679

680 Zijun Shen, Naohao Huang, Fanyi Yang, Yutong Wang, Guoxiong Gao, Tianyi Xu, Jiedong Jiang,
 681 Wanyi He, Pu Yang, Mengzhou Sun, Haocheng Ju, Peihao Wu, Bryan Dai, and Bin Dong.
 682 REAL-Prover: Retrieval augmented Lean prover for mathematical reasoning. *arXiv preprint*
arXiv:2505.20613, 2025.

683

684 Yuda Song, Hanlin Zhang, Carson Eisenach, Sham M. Kakade, Dean Foster, and Udaya Ghai. Mind
 685 the gap: Examining the self-improvement capabilities of large language models. In *The Thirteenth*
686 International Conference on Learning Representations, 2025.

687

688 Kaya Stechly, Karthik Valmeekam, and Subbarao Kambhampati. On the self-verification limita-
 689 tions of large language models on reasoning and planning tasks. In *The Thirteenth International*
690 Conference on Learning Representations, 2025.

691

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

694

695 George Tsoukalas, Jasper Lee, John Jennings, Jimmy Xin, Michelle Ding, Michael Jennings, Ami-
 696 tayush Thakur, and Swarat Chaudhuri. Putnambench: Evaluating neural theorem-provers on the
 697 putnam mathematical competition. In *Advances in Neural Information Processing Systems*, vol-
 ume 37, pp. 11545–11569, 2024a.

698

699 George Tsoukalas, Jasper Lee, John Jennings, Jimmy Xin, Michelle Ding, Michael Jennings, Ami-
 700 tayush Thakur, and Swarat Chaudhuri. PutnamBench: Evaluating neural theorem-provers on the
 701 putnam mathematical competition. In *The Thirty-eighth Conference on Neural Information Pro-*
cessing Systems Datasets and Benchmarks Track, 2024b.

702 Haiming Wang, Ye Yuan, Zhengying Liu, Jianhao Shen, Yichun Yin, Jing Xiong, Enze Xie, Han
 703 Shi, Yujun Li, Lin Li, Jian Yin, Zhenguo Li, and Xiaodan Liang. DT-solver: Automated theorem
 704 proving with dynamic-tree sampling guided by proof-level value function. In *Proceedings of the*
 705 *61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*,
 706 pp. 12632–12646. Association for Computational Linguistics, 2023. doi: 10.18653/v1/2023.
 707 acl-long.706.

708 Haiming Wang, Huajian Xin, Zhengying Liu, Wenda Li, Yinya Huang, Jianqiao Lu, Zhicheng Yang,
 709 Jing Tang, Jian Yin, Zhenguo Li, and Xiaodan Liang. Proving theorems recursively. In *Advances*
 710 *in Neural Information Processing Systems*, volume 37, pp. 86720–86748, 2024a.

711 Haiming Wang, Mert Unsal, Xiaohan Lin, Mantas Baksys, Junqi Liu, Marco Dos Santos, Flood
 712 Sung, Marina Vinyes, Zhenzhe Ying, Zekai Zhu, Jianqiao Lu, Hugues de Saxcé, Bolton Bailey,
 713 Chendong Song, Chenjun Xiao, Dehao Zhang, Ebony Zhang, Frederick Pu, Han Zhu, Jiawei Liu,
 714 Jonas Bayer, Julien Michel, Longhui Yu, Léo Dreyfus-Schmidt, Lewis Tunstall, Luigi Pagani,
 715 Moreira Machado, Pauline Bourigault, Ran Wang, Stanislas Polu, Thibaut Baroyer, Wen-Ding
 716 Li, Yazhe Niu, Yann Fleureau, Yangyang Hu, Zhouliang Yu, Zihan Wang, Zhilin Yang, Zhengying
 717 Liu, and Jia Li. Kimina-prover preview: Towards large formal reasoning models with reinforce-
 718 ment learning. *arXiv preprint arXiv:2504.11354*, 2025.

719 Ruida Wang, Jipeng Zhang, Yizhen Jia, Rui Pan, Shizhe Diao, Renjie Pi, and Tong Zhang. The-
 720 remLlama: Transforming general-purpose LLMs into lean4 experts. In *Proceedings of the 2024*
 721 *Conference on Empirical Methods in Natural Language Processing*, pp. 11953–11974. Associa-
 722 tion for Computational Linguistics, 2024b. doi: 10.18653/v1/2024.emnlp-main.667.

723 Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, brian ichter, Fei Xia, Ed Chi, Quoc V
 724 Le, and Denny Zhou. Chain-of-thought prompting elicits reasoning in large language models. In
 725 *Advances in Neural Information Processing Systems*, volume 35, pp. 24824–24837, 2022.

726 Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi,
 727 Pierrick Cistac, Tim Rault, Remi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick
 728 von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger,
 729 Mariama Drame, Quentin Lhoest, and Alexander Rush. Transformers: State-of-the-art natural
 730 language processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural*
 731 *Language Processing: System Demonstrations*, pp. 38–45. Association for Computational Lin-
 732 giistics, 2020.

733 Minchao Wu, Michael Norrish, Christian Walder, and Amir Dezfouli. Tacticzero: Learning to prove
 734 theorems from scratch with deep reinforcement learning. In *Advances in Neural Information*
 735 *Processing Systems*, volume 34, pp. 9330–9342, 2021.

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

739 Zijian Wu, Jiayu Wang, Dahua Lin, and Kai Chen. LEAN-GitHub: Compiling github lean reposi-
 740 tories for a versatile lean prover. *arXiv preprint arXiv:2407.17227*, 2024b.

741 Huajian Xin, Z. Z. Ren, Junxiao Song, Zhihong Shao, Wanjia Zhao, Haocheng Wang, Bo Liu,
 742 Liyue Zhang, Xuan Lu, Qiushi Du, Wenjun Gao, Qihao Zhu, Dejian Yang, Zhibin Gou, Z. F.
 743 Wu, Fuli Luo, and Chong Ruan. DeepSeek-Prover-V1.5: Harnessing proof assistant feedback
 744 for reinforcement learning and monte-carlo tree search. *International Conference on Learning*
 745 *Representations*, 2025a.

746 Ran Xin, Chenguang Xi, Jie Yang, Feng Chen, Hang Wu, Xia Xiao, Yifan Sun, Shen Zheng, and
 747 Kai Shen. BFS-Prover: Scalable best-first tree search for llm-based automatic theorem proving.
 748 *arXiv preprint arXiv:2502.03438*, 2025b.

749 Ziwei Xu, Sanjay Jain, and Mohan Kankanhalli. Hallucination is inevitable: An innate limitation of
 750 large language models. *arXiv preprint arXiv:2401.11817*, 2025.

756 An Yang, Beichen Zhang, Binyuan Hui, Bofei Gao, Bowen Yu, Chengpeng Li, Dayiheng Liu,
 757 Jianhong Tu, Jingren Zhou, Junyang Lin, Keming Lu, Mingfeng Xue, Runji Lin, Tianyu Liu,
 758 Xingzhang Ren, and Zhenru Zhang. Qwen2.5-Math technical report: Toward mathematical ex-
 759 pert model via self-improvement. *arXiv preprint arXiv:2409.12122*, 2024.

760 An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang
 761 Gao, Chengan Huang, Chenxu Lv, Chujie Zheng, Dayiheng Liu, Fan Zhou, Fei Huang, Feng Hu,
 762 Hao Ge, Haoran Wei, Huan Lin, Jialong Tang, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin
 763 Yang, Jiaxi Yang, Jing Zhou, Jingren Zhou, Junyang Lin, Kai Dang, Keqin Bao, Kexin Yang,
 764 Le Yu, Lianghao Deng, Mei Li, Mingfeng Xue, Mingze Li, Pei Zhang, Peng Wang, Qin Zhu, Rui
 765 Men, Ruize Gao, Shixuan Liu, Shuang Luo, Tianhao Li, Tianyi Tang, Wenbiao Yin, Xingzhang
 766 Ren, Xinyu Wang, Xinyu Zhang, Xuancheng Ren, Yang Fan, Yang Su, Yichang Zhang, Yinger
 767 Zhang, Yu Wan, Yuqiong Liu, Zekun Wang, Zeyu Cui, Zhenru Zhang, Zhipeng Zhou, and Zihan
 768 Qiu. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*, 2025a.

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

773 Kaiyu Yang, Gabriel Poesia, Jingxuan He, Wenda Li, Kristin E. Lauter, Swarat Chaudhuri, and
 774 Dawn Song. Position: Formal mathematical reasoning—a new frontier in AI. In *Forty-second
 775 International Conference on Machine Learning Position Paper Track*, 2025b.

776 Huaiyuan Ying, Zijian Wu, Yihan Geng, Jiayu Wang, Dahua Lin, and Kai Chen. Lean workbook:
 777 A large-scale lean problem set formalized from natural language math problems. In *Advances in
 778 Neural Information Processing Systems*, volume 37, pp. 105848–105863, 2024.

779 Roozbeh Yousefzadeh and Xuenan Cao. A lean dataset for international math olympiad: Small steps
 780 towards writing math proofs for hard problems. *Transactions on Machine Learning Research*,
 781 2025. ISSN 2835-8856.

782 Zhiyuan Zeng, Qinyuan Cheng, Zhangyue Yin, Yunhua Zhou, and Xipeng Qiu. Revisiting the
 783 test-time scaling of o1-like models: Do they truly possess test-time scaling capabilities? *arXiv
 784 preprint arXiv:2502.12215*, 2025.

785 Jingyuan Zhang, Qi Wang, Xinguang Ji, Yahui Liu, Yang Yue, Fuzheng Zhang, Di Zhang, Guorui
 786 Zhou, and Kun Gai. Leanabell-Prover: Posttraining scaling in formal reasoning. *arXiv preprint
 787 arXiv:2504.06122*, 2025.

788 Kunhao Zheng, Jesse Michael Han, and Stanislas Polu. miniF2F: a cross-system benchmark for
 789 formal olympiad-level mathematics. In *International Conference on Learning Representations*,
 790 2022.

791 Andy Zhou, Kai Yan, Michal Shlapentokh-Rothman, Haohan Wang, and Yu-Xiong Wang. Language
 792 agent tree search unifies reasoning, acting, and planning in language models. In *Proceedings of
 793 the 41st International Conference on Machine Learning*, volume 235, pp. 62138–62160. PMLR,
 794 2024.

795 Yichi Zhou, Jianqiu Zhao, Yongxin Zhang, Bohan Wang, Siran Wang, Luoxin Chen, Jiahui Wang,
 796 Huawei Chen, Allan Jie, Xinbo Zhang, Haocheng Wang, Luong Trung, Rong Ye, Phan Nhat
 797 Hoang, Huishuai Zhang, Peng Sun, and Hang Li. Solving formal math problems by decomposi-
 798 tion and iterative reflection. *arXiv preprint arXiv:2507.15225*, 2025.

799
 800
 801
 802
 803
 804
 805
 806
 807
 808
 809

810 A EXTENDED RELATED WORK
811812 We briefly summarized related work in Section 2. Here we provide details of representative systems.
813814 A.1 LANGUAGE MODELS FOR FORMAL THEOREM PROVING
815816 The use of language models for guiding formal theorem provers has gained momentum recently.
817 Early work like GPT-f (Polu & Sutskever, 2020) applied transformers to produce proofs in formal
818 systems, such as Metamath (Megill & Wheeler, 2019) and Lean (Moura & Ullrich, 2021), by gen-
819 erating one proof step (tactic) at a time, guided by a goal state. Subsequent efforts in Lean, such as
820 lean-gpt⁴ and PACT (Han et al., 2022), fine-tuned LLMs on large corpora of proof data, achieving
821 moderate success in automatically discovering proofs.
822823 A.2 TREE-SEARCH-BASED FORMAL PROVING
824825 BFS-Prover (Xin et al., 2025b) proposed a scalable best-first tree search framework for Lean 4 that
826 incorporates three key innovations: strategic data filtering during expert iterations, direct preference
827 optimization (DPO) (Rafailov et al., 2023) on state-tactic pairs using Lean compiler feedback, and
828 length normalization to encourage exploration of deeper proof paths. InternLM2.5-StepProver (Wu
829 et al., 2024a) combined expert iteration with BFS and critic-guided sampling, while Hunyuan-
830 Prover (Li et al., 2025) integrated large-scale data synthesis and guided search. Reinforcement-
831 enhanced variants such as DeepSeek-Prover-V1.5 (Xin et al., 2025a) proposed the use of RMaxTS,
832 a variant of Monte-Carlo tree search (MCTS), to diversify exploration and improve success rates.
833

834 A.3 WHOLE-PROOF GENERATION

835 Representative systems in this strand have advanced two complementary mechanisms: (i) expert-
836 iteration bootstrapping, which cycles model-generated proofs through a formal verifier to curate
837 training trajectories, and (ii) reinforcement learning (RL) with verifier feedback that directly optimizes
838 long, one-shot scripts (often with a long chain-of-thought).
839840 Polu et al. (2023) introduced expert iteration for formal mathematics, alternating proof search with
841 learning. They showed expert iteration outperforms search-only at fixed compute, discovered an
842 automatically paced curriculum from problem statements, and showed improved performance on
843 the miniF2F (Zheng et al., 2022) benchmark without requiring ground-truth proofs. InternLM2.5-
844 StepProver (Wu et al., 2024a) scaled expert iteration on Lean-Workbook (Ying et al., 2024), trained a
845 critic to prioritize easier instances and guide deeper proofs, and paired expert iteration with best-first
846 exploration, achieving strong results on several benchmarks, such as miniF2F (Zheng et al., 2022),
847 ProofNet (Azerbayev et al., 2023), PutnamBench (Tsoukalas et al., 2024a), and Lean-Workbook-
848 Plus (Ying et al., 2024). Lean-STaR (Lin et al., 2025a) trained a model to interleave informal
849 natural-language thoughts with formal tactic steps. The model is trained by expert iteration, and
850 at inference time, it generates informal reasoning prior to each tactic, enhancing theorem-proving
851 performance. Goedel-Prover (Lin et al., 2025b) tackled data scarcity by training statement formal-
852 izers to translate Numina problems into Lean 4, building a 1.64M-statement corpus, and iteratively
853 bootstrapping provers whose new proofs are added to training. The resulting SFT-centered expert
854 iteration pipeline surpasses prior open-source baselines. Goedel-Prover-V2 (Lin et al., 2025c) ex-
855 tends expert iteration with scaffolded data synthesis, verifier-guided self-correction, and model
856 averaging, delivering large gains on the MiniF2F benchmark (Zheng et al., 2022) at 8–32B scales
857 under constrained test-time budgets.
858859 Kaliszyk et al. (2018) formulated theorem proving as reinforcement learning for connection-style
860 proof search, using Monte Carlo simulations guided by rewards from previous attempts to re-
861 place hand-crafted heuristics and improve held-out performance. DeepSeek-Prover-V1.5 (Xin et al.,
862 2025a) utilized reinforcement learning from proof assistant feedback (RLPAF) and a novel Monte-
863 Carlo tree search variant, RMaxTS, which employs an intrinsic-reward-driven strategy to explore
864 diverse proof paths. Leanabell-Prover (Zhang et al., 2025) demonstrated the effectiveness of post-
865 training in formal theorem proving by applying continual training with data emulating human cog-
866867 ⁴<https://github.com/jesse-michael-han/lean-gptf>

nitive behaviors and reinforcement learning with compiler feedback to existing models. Kimina-Prover Preview (Wang et al., 2025) employed a large-scale reinforcement learning pipeline and a structured “formal reasoning pattern,” emulating human problem-solving strategies. It achieves an 80.7% pass rate on MiniF2F (Zheng et al., 2022) with a 72B-parameter model. Leanabell-Prover-V2 (Ji et al., 2025) is built on Kimina-Prover-Preview-Distill-7B(Wang et al., 2025) and DeepSeek-Prover-V2-7B (Ren et al., 2025) as base models, and further improved through post-training with reinforcement learning.

872 A.4 FORMAL THEOREM PROVING WITH RETRIEVAL-AUGMENTED GENERATION

874 Retrieval-augmented provers query large formal libraries at inference time and condition generation
 875 on the retrieved items, typically relevant lemmas, theorems, or proof patterns from mathlib (math-
 876 lib Community, 2020). This mitigates the limits of parametric memory by injecting on-demand
 877 knowledge and can be applied to both stepwise tactic generation and whole-proof scripts. Lean-
 878 Dojo (Yang et al., 2023) established the core infrastructure for RAG in Lean, including fine-grained
 879 premise annotations, a gym-like interactive environment, and a retrieval-augmented prover that se-
 880 lects premises for each proof state. REAL-Prover (Shen et al., 2025) integrated a semantic premise
 881 selector (LeanSearch-PS) with a fine-tuned Lean 4 prover and reports gains on challenging bench-
 882 marks such as ProofNet (Azerbayev et al., 2023).

883 A.5 PROOF REFINEMENT AND SUBGOAL DECOMPOSITION

884 Jiang et al. (2023) introduced Draft, Sketch, and Prove (DSP), a novel three-stage method that lever-
 885 ages informal proofs to guide automated theorem provers. The process involves drafting an informal
 886 proof (either by a human or an LLM), using a language model to convert it into a high-level formal
 887 sketch with verifiable steps, and finally employing an off-the-shelf prover to automatically solve
 888 the remaining logical gaps. This approach of guiding a formal prover with an informal-to-formal
 889 sketch significantly improved its success rate, boosting performance on the miniF2F benchmark
 890 from 20.9% to 39.3%.

891 Wang et al. (2024a) introduced POETRY, a novel method that proves theorems recursively to over-
 892 come the limitations of short-sighted, step-by-step search in automated theorem proving. By first
 893 finding a verifiable high-level proof sketch and deferring detailed sub-proofs to subsequent lev-
 894 els using a *sorry* tactic, POETRY can solve more complex problems and find significantly longer
 895 proofs, leading to superior results on the miniF2F (Zheng et al., 2022) and PISA (Jiang et al., 2021)
 896 benchmarks.

897 Cao et al. (2025) introduced DSP+, an improved Draft, Sketch, and Prove framework Jiang et al.
 898 (2023) that achieves high performance in automated theorem proving without requiring any model
 899 training or fine-tuning. By carefully coordinating existing off-the-shelf reasoning models and step
 900 provers with fine-grained neuro-symbolic enhancements at each stage, DSP+ solved 80.7% of the
 901 miniF2F benchmark (Zheng et al., 2022), which was comparable to top models that rely on extensive
 902 reinforcement learning, and even proved a previously unsolved IMO problem.

903 DeepSeek-Prover-V2 (Ren et al., 2025) used a powerful general-purpose model, DeepSeek-
 904 V3 (DeepSeek-AI, 2024), to break down complex theorems into simpler subgoals, which are then
 905 recursively solved and synthesized into a cold-start dataset for the final prover. The resulting model
 906 achieved an 88.9% pass rate on the MiniF2F benchmark (Zheng et al., 2022).

907 Delta Prover (Zhou et al., 2025) is an agent-based framework that enables a general-purpose LLM to
 908 solve formal math problems without any specialized fine-tuning. The agent orchestrated the LLM’s
 909 interaction with the Lean 4 environment through a novel process of reflective decomposition and
 910 iterative proof repair, where the model breaks down complex problems and corrects its own errors
 911 based on compiler feedback. This training-free approach achieved a 95.9% success rate on the
 912 miniF2F benchmark (Zheng et al., 2022), surpassing all previous methods, including those requiring
 913 extensive specialized training.

914 Chen et al. (2025) introduced Seed-Prover, a whole-proof reasoning model that uses a novel lemma-
 915 style approach to solve complex formal math problems. Seed-Prover iteratively refined its proofs
 916 using compiler feedback and a shared pool of proved lemmas, employing a powerful three-tiered
 917 test-time inference strategy for both deep and broad reasoning. This method significantly surpassed

918 **Algorithm 1** The overall architecture of our lemma-based theorem-proving agent coordinating in-
 919 formal reasoning, formal reasoning, and Lean.
 920

921 **Input:** Problem T with hyperparameters N_{init} (max initial proof attempts) and N_{refine} (max refinement attempts)
 922 **Output:** Formal proof of T or *failure*

923 **function** $\text{MAIN}(T)$: Overall proof process for problem T

924 $P_{\text{direct}} \leftarrow \text{PROVE}(T)$: Attempt to prove theorem T directly

925 **if** P_{direct} succeeds **then**

926 **return** P_{direct}

927 **end if**

928 // Generate lemmas

929 Informal LLM generates lemmas L_1, L_2, \dots, L_n in natural
 930 language

931 **for** each lemma L_i **do**

932 AutoFormalizer converts L_i into Lean statement F_i

933 Lean checks F_i . If failing, regenerate F_i until syntactically
 934 correct

935 **end for**

936 // Prove each lemma

937 **for** each lemma F_i **do**

938 $P_i \leftarrow \text{PROVE}(F_i)$: Attempt to prove lemma F_i

939 **end for**

940 // Collect proven lemmas

941 $\mathcal{P}_{\text{proven}} \leftarrow \{P_i \mid P_i \text{ is succeeded}\}$

942 // Synthesize final proof using proven lemmas

943 **for** $k = 1$ to N_{init} **do**

944 $P_{\text{final}} \leftarrow$ Prover synthesizes proof of T using $\mathcal{P}_{\text{proven}}$

945 Lean checks P_{final}

946 **if** the check succeeds **then**

947 **return** P_{final}

948 **end if**

949 **end for**

950 // Iterative refinement of final proof

951 $P_{\text{best}} \leftarrow$ Best previous proof attempt with the fewest Lean
 952 errors

953 **return** $\text{ITERATIVEREFINE}(P_{\text{best}})$

954 **end function**

955

956 all previous state-of-the-art results, saturating the MiniF2F benchmark (Zheng et al., 2022), proving
 957 78.1% of past IMO problems, and solving 5 out of 6 problems at the IMO 2025 competition.

958

959 **B PSEUDOCODE OF THE OVERALL WORKFLOW**

960

961 The pseudocode of our overall workflow is shown in Algorithm 1.

962

963 **C DETAILED THEORETICAL ANALYSIS**

964

965 We briefly discussed the theoretical analysis of our approach in Section 4. In this section, we provide
 966 a detailed theoretical analysis of our approach.

967

968 **C.1 BENEFITS OF LEMMAS FOR STRUCTURED PROOF DECOMPOSITION**

969

970 We begin by stating a lemma required for the following analysis:

971 **Lemma C.1** (Number of Trials for Success). *Let p denote the probability that the model successfully
 972 proves a theorem T . Then the expected number of trials until the first success, N , and the number
 973 of trials required to succeed with probability at least $1 - \delta$, denoted N_δ , satisfy the following:*

974
$$\mathbb{E}[N] = \frac{1}{p}, \quad \log(1/\delta) \left(\frac{1}{p} - 1 \right) < \frac{\log \delta}{\log(1-p)} < N_\delta = \left\lceil \frac{\log \delta}{\log(1-p)} \right\rceil < \frac{\log(1/\delta)}{p} + 1.$$

972 *Proof.* Since each trial is an independent Bernoulli experiment with success probability p , the number of trials N until the first success follows a geometric distribution. It is well known that
 973
 974

$$975 \quad 976 \quad 977 \quad \mathbb{E}[N] = \sum_{n=1}^{\infty} n(1-p)^{n-1}p = \frac{1}{p}.$$

978 Next, we consider N_{δ} . Since the probability of at least one success in n trials is $1 - (1-p)^n$, the
 979 condition for achieving success with probability at least $1 - \delta$ is:
 980

$$981 \quad 982 \quad 983 \quad 1 - (1-p)^n = 1 - \delta \Leftrightarrow (1-p)^n = \delta \Leftrightarrow n = \frac{\log \delta}{\log(1-p)}.$$

984 Recalling the standard inequalities $p \leq -\log(1-p) \leq \frac{p}{1-p}$, which is valid for $0 < p < 1$, together
 985 with the basic ceiling inequality $x \leq \lceil x \rceil < x + 1$, we obtain:
 986

$$987 \quad 988 \quad \log(1/\delta) \left(\frac{1}{p} - 1 \right) < \frac{\log \delta}{\log(1-p)} < N_{\delta} = \left\lceil \frac{\log \delta}{\log(1-p)} \right\rceil < \frac{\log(1/\delta)}{p} + 1.$$

989 This completes the proof. \square
 990

991 For simplicity, we henceforth relax N_{δ} to be continuous and write:
 992

$$993 \quad 994 \quad \log(1/\delta) \left(\frac{1}{p} - 1 \right) < N_{\delta} = \frac{\log \delta}{\log(1-p)} < \frac{\log(1/\delta)}{p}.$$

995 The difference from the actual integer-valued N_{δ} is at most less than 1.
 996

997 As rigorous versions of Theorems 4.4 to 4.5 described in Section 4.1, we obtain the following
 998 Theorems C.2 to C.3, under the same Assumptions 4.1 to 4.3:
 999

1000 **Theorem C.2** (Required Number of Trials). *Let N_{dir} denote the number of trials required to directly
 1001 prove a problem T with probability at least $1 - \delta$. Let N_{lem} denote the total number of trials required
 1002 to complete the proof of T with probability at least $1 - \delta$, when lemmas L_1, \dots, L_n are introduced
 1003 with an allowed failure probability δ_{lem} . Suppose each lemma L_i contains a subset of the essential
 1004 intermediate facts $\{F_i\}_{i \in S_i}$ with $S_i \subseteq [m]$. Then the following holds:*

$$1004 \quad \Phi_{\text{dir}}(p) - \log(1/\delta) < N_{\text{dir}} < \Phi_{\text{dir}}(p), \\ 1005 \quad \Phi_{\text{lem}}(p) - \log(1/\delta) - n \log(1/\delta_{\text{lem}}) < \mathbb{E}[N_{\text{lem}}] < \Phi_{\text{lem}}(p),$$

1006 where
 1007

$$1008 \quad \Phi_{\text{dir}}(p) := \log(1/\delta) \prod_{i=1}^m \frac{1}{p_i},$$

$$1009 \quad \Phi_{\text{lem}}(p) := \log(1/\delta_{\text{lem}}) \sum_{i=1}^n \prod_{j \in S_i} \frac{1}{p_j} + \frac{\log(1/\delta)}{r_0} \left(\prod_{i \in R_0} \frac{1}{p_i} \right) \prod_{i=1}^n \left((1 - \delta_{\text{lem}}) + \delta_{\text{lem}} \prod_{j \in S_i} \frac{1}{p_j} \right).$$

1010 Here, we denote $U := \bigcup_{i=1}^n S_i$, $R_0 := [m] \setminus U$, and $r_0 := \min P(F_S | \{F_i\}_{i \in S})$.
 1011

1012 *Proof.* By Assumption 4.2, the probability that all F_1, \dots, F_m succeed and the problem T is solved
 1013 equals $\prod_{i=1}^m p_i$. Hence, by Lemma C.1, we obtain:
 1014

$$1015 \quad \Phi_{\text{dir}}(p) - \log(1/\delta) < N_{\text{dir}} < \Phi_{\text{dir}}(p).$$

1016 Similarly, since the probability that all F_j with $j \in S_i$ succeed and lemma L_i is proved equals
 1017 $\prod_{j \in S_i} p_j$, the number of trials required for lemma L_i , denoted N_{L_i} , satisfies:
 1018

$$1019 \quad \log(1/\delta_{\text{lem}}) \prod_{j \in S_i} \frac{1}{p_j} - \log(1/\delta_{\text{lem}}) < N_{L_i} < \log(1/\delta_{\text{lem}}) \prod_{j \in S_i} \frac{1}{p_j}.$$

1026 Therefore, the total number of trials required to prove all n lemmas L_1, \dots, L_n is bounded by the
 1027 sum of the bounds above, i.e.,
 1028

$$1029 \log(1/\delta_{\text{lem}}) \sum_{i=1}^n \prod_{j \in S_i} \frac{1}{p_j} - n \log(1/\delta_{\text{lem}}) < \sum_{i=1}^n N_{L_i} < \log(1/\delta_{\text{lem}}) \sum_{i=1}^n \prod_{j \in S_i} \frac{1}{p_j}. \quad (1)$$

1032 The probability that the composition of all lemmas succeeds is r_0 , while the probability of proving
 1033 the uncovered facts $\{F_i\}_{i \in R_0}$ is $\prod_{i \in R_0} p_i$. If a lemma L_i fails with probability δ_{lem} , then in the final
 1034 proof it must be reproved directly, which succeeds with probability $\prod_{j \in S_i} p_j$. Thus, the expected
 1035 success probability of lemma L_i in the final stage is: $(1 - \delta_{\text{lem}}) + \delta_{\text{lem}} \prod_{j \in S_i} p_j$.
 1036

1037 Therefore, since the expected success probability in the final stage is given by the product above, the
 1038 number of trials required to complete the proof of the whole problem T using lemmas in the final
 1039 stage, denoted N_{final} , satisfies:
 1040

$$1041 \Phi_{\text{final}}(p) - \log(1/\delta) < \mathbb{E}[N_{\text{final}}] < \Phi_{\text{final}}(p), \quad (2)$$

1043 where

$$1044 \Phi_{\text{final}}(p) := \frac{\log(1/\delta)}{r_0} \left(\prod_{i \in R_0} \frac{1}{p_i} \right) \prod_{i=1}^n \left((1 - \delta_{\text{lem}}) + \delta_{\text{lem}} \prod_{j \in S_i} \frac{1}{p_j} \right).$$

1045 Hence, by combining Equations (1) and (2), we obtain the desired result, completing the proof of
 1046 Theorem C.2. \square
 1047

1048 From Theorem C.2, we see that decomposing the problem into lemmas transforms the corresponding
 1049 leading term from a product into a sum, thereby significantly reducing the order of the required
 1050 number of trials.
 1051

1052 **Theorem C.3** (Threshold Condition for Lemma Efficiency). *There exists a threshold $\tau \in [0, 1]$ such
 1053 that if $p_i \leq \tau$ for all $i \in [m]$, then $\mathbb{E}[N_{\text{lem}}] \leq N_{\text{dir}}$ holds for any $\delta, \delta_{\text{lem}} \in (0, 1)$.*
 1054

1055 *Proof.* Consider the condition $\frac{\mathbb{E}[N_{\text{lem}}]}{N_{\text{dir}}} < 1$. By Theorem C.2, this condition is satisfied if the
 1056 following holds:
 1057

$$1058 \frac{\Phi_{\text{lem}}(p)}{\Phi_{\text{dir}}(p) - \log(1/\delta)} < 1 \\ 1059 \Leftrightarrow \frac{\log(1/\delta_{\text{lem}}) \sum_{i=1}^n \prod_{j \in S_i} \frac{1}{p_j}}{\log(1/\delta) \prod_{i=1}^m \left(\frac{1}{p_i} - 1 \right)} \\ 1060 + \frac{\frac{\log(1/\delta)}{r_0} \left(\prod_{i \in R_0} \frac{1}{p_i} \right) \prod_{i=1}^n \left((1 - \delta_{\text{lem}}) + \delta_{\text{lem}} \prod_{j \in S_i} \frac{1}{p_j} \right)}{\log(1/\delta) \prod_{i=1}^m \left(\frac{1}{p_i} - 1 \right)} < 1. \quad (3)$$

1061 The first term on the left-hand side (LHS) of Equation (3) can be rewritten as:
 1062

$$1063 \frac{\log(1/\delta_{\text{lem}}) \sum_{i=1}^n \prod_{j \in S_i} \frac{1}{p_j}}{\log(1/\delta) \prod_{i=1}^m \left(\frac{1}{p_i} - 1 \right)} = \frac{\log(1/\delta_{\text{lem}})}{\log(1/\delta)} \sum_{i=1}^n \frac{\prod_{j \in S_i} \frac{1}{p_j} \prod_{j=1}^m p_j}{1 - \prod_{j=1}^m p_j} \\ 1064 = \frac{\log(1/\delta_{\text{lem}})}{\log(1/\delta)} \sum_{i=1}^n \frac{\prod_{j \notin S_i} p_j}{1 - \prod_{j=1}^m p_j}. \quad (4)$$

1080 The second term on the LHS of Equation (3) can be rewritten as:
1081

$$\begin{aligned}
& \frac{\log(1/\delta)}{r_0} \left(\prod_{i \in R_0} \frac{1}{p_i} \right) \prod_{i=1}^n \left((1 - \delta_{\text{lem}}) + \delta_{\text{lem}} \prod_{j \in S_i} \frac{1}{p_j} \right) \\
& \quad \log(1/\delta) \prod_{i=1}^m \left(\frac{1}{p_i} - 1 \right) \\
& = \frac{1}{r_0} \left(\prod_{i \in R_0} \frac{1}{p_i} \right) \left(\prod_{i=1}^n \left((1 - \delta_{\text{lem}}) + \delta_{\text{lem}} \prod_{j \in S_i} \frac{1}{p_j} \right) \right) \frac{\prod_{j=1}^m p_j}{1 - \prod_{j=1}^m p_j} \\
& = \frac{1}{r_0} \prod_{i=1}^n \left((1 - \delta_{\text{lem}}) \prod_{j \in S_i} p_j + \delta_{\text{lem}} \right) \frac{1}{1 - \prod_{j=1}^m p_j}. \tag{5}
\end{aligned}$$

1092 From Equations (4) and (5), both the first and second terms on the LHS of Equation (3) are monotonically
1093 increasing with respect to p_i . Hence, the LHS of Equation (3) itself is monotonically increasing
1094 w.r.t. p_i . Therefore, by bounding the LHS of Equation (3) from above by using $p_{\max} := \max_i p_i$
1095 and solving for p_{\max} , we obtain a sufficient condition, completing the proof. \square
1096

1097 From Theorem C.3, it follows that lemma generation is effective for difficult problems. Therefore,
1098 our strategy of generating lemmas for difficult problems and solving easy problems directly is justified.
1099

1100 **Theorem C.4** (Optimal Partition of Lemma Coverage). *Under the fixed lemma coverage $U :=$*
1101 $\bigcup_{i=1}^n S_i \subseteq [m]$, $\mathbb{E}[N_{\text{lem}}]$ *is minimized when $\log p(S_i)$ is as close as possible to $\frac{1}{n} \log p(U)$ for all*
1102 $i \in [n]$, *where $p(S_i) := \prod_{j \in S_i} p_j$ and $p(U) := \prod_{j \in U} p_j$.*
1103

1104 *Proof.* From Theorem C.2, we consider minimizing $\Phi_{\text{lem}}(p)$. Let $W := \prod_{i \in U} \frac{1}{p_i}$.
1105

1106 By Jensen's inequality, the first term of $\Phi_{\text{lem}}(p)$ can be bounded as follows:
1107

$$\begin{aligned}
\log(1/\delta_{\text{lem}}) \sum_{i=1}^n \prod_{j \in S_i} \frac{1}{p_j} &= \log(1/\delta_{\text{lem}}) \sum_{i=1}^n \exp\left(\sum_{j \in S_i} \log \frac{1}{p_j}\right) \\
&\geq \log(1/\delta_{\text{lem}}) n \exp\left(\frac{1}{n} \sum_{i=1}^n \sum_{j \in S_i} \log \frac{1}{p_j}\right) \\
&= \log(1/\delta_{\text{lem}}) n \exp\left(\frac{1}{n} \log W\right)
\end{aligned}$$

1116 with equality if and only if $\log p(S_i) = \frac{1}{n} \log p(U)$ for all $i \in [n]$.
1117

1118 Noting that $f(x) = \log((1-d)+d \exp(x))$ is convex for $d \in (0, 1)$, we can apply Jensen's inequality
1119 to bound the second term of $\Phi_{\text{lem}}(p)$ as follows:
1120

$$\begin{aligned}
& \frac{\log(1/\delta)}{r_0} \left(\prod_{i \in R_0} \frac{1}{p_i} \right) \prod_{i=1}^n \left((1 - \delta_{\text{lem}}) + \delta_{\text{lem}} \prod_{j \in S_i} \frac{1}{p_j} \right) \\
&= \frac{\log(1/\delta)}{r_0} \left(\prod_{i \in R_0} \frac{1}{p_i} \right) \exp\left(\sum_{i=1}^n \log\left((1 - \delta_{\text{lem}}) + \delta_{\text{lem}} \exp\left(\sum_{j \in S_i} \log \frac{1}{p_j}\right)\right)\right) \\
&\geq \frac{\log(1/\delta)}{r_0} \left(\prod_{i \in R_0} \frac{1}{p_i} \right) \exp\left(n \log\left((1 - \delta_{\text{lem}}) + \delta_{\text{lem}} \exp\left(\frac{1}{n} \sum_{i=1}^n \sum_{j \in S_i} \log \frac{1}{p_j}\right)\right)\right) \\
&= \frac{\log(1/\delta)}{r_0} \left(\prod_{i \in R_0} \frac{1}{p_i} \right) \exp\left(n \log\left((1 - \delta_{\text{lem}}) + \delta_{\text{lem}} \exp\left(\frac{1}{n} \log W\right)\right)\right)
\end{aligned}$$

1133 with equality if and only if $\log p(S_i) = \frac{1}{n} \log p(U)$ for all $i \in [n]$.
1134

1134 Therefore, since both the first and second terms of $\Phi_{\text{lem}}(p)$ attain their minimum under the same
 1135 condition, namely:

$$1136 \log p(S_i) = \frac{1}{n} \log p(U) \quad \text{for all } i \in [n],$$

1138 it follows that $\Phi_{\text{lem}}(p)$ itself is minimized under this condition. In the discrete case, the minimum is
 1139 achieved at the partition closest to this balanced condition. This completes the proof. \square
 1140

1141 Theorem C.4 suggests that the optimal lemmas are those that divide the problem into subproblems
 1142 of approximately equal difficulty.
 1143

1144 C.2 BENEFITS OF LEMMAS FOR DISCOVERING PROOF STRATEGIES (E.G., SPECIAL CASES)

1146 **Theorem C.5** (Success Probability Improvement by Lemmas (Restated)). *The success probability
 1147 of performing one trial of final proving by sampling a strategy from the posterior distribution π_n is
 1148 bounded as follows:*

$$1149 \mathbb{E}[\mathbb{P}(\text{succ@1})] \geq r \exp(-H_0 + I(Z; Y_{1:n})).$$

1151 *Proof.* We begin with:

$$1153 \mathbb{P}(\text{succ@1} \mid Z = z, Y = y) = p(z) \pi(z \mid y).$$

1154 Taking expectation, we obtain:

$$1156 \mathbb{E}_{Z,Y}[\mathbb{P}(\text{succ@1} \mid Z, Y)] = \mathbb{E}_{Z,Y}[p(Z) \pi(Z \mid Y)] \\ 1157 = \mathbb{E}_{Z,Y}[p(Z) \pi_n(Z)] \\ 1158 \geq r \mathbb{E}_{Z,Y}[\pi_n(Z)]. \quad (6)$$

1160 It remains to lower-bound $\mathbb{E}_{Z,Y}[\pi_n(Z)]$.

1161 For fixed $Y = y$, we have:

$$1163 \mathbb{E}_Z[\pi_n(Z) \mid Y = y] = \sum_{z \in \mathcal{S}} \pi_n(z) \mathbb{P}(Z = z \mid Y = y) \\ 1164 = \sum_{z \in \mathcal{S}} \pi_n(z)^2.$$

1168 Taking expectation over Y yields:

$$1170 \mathbb{E}_{Z,Y}[\pi_n(Z)] = \mathbb{E}_Y \left[\mathbb{E}_Z[\pi_n(Z) \mid Y] \right] = \mathbb{E}_Y \left[\sum_{z \in \mathcal{S}} \pi_n(z)^2 \right].$$

1172 By Lemma C.6, we have:

$$1174 \sum_{z \in \mathcal{S}} \pi(z \mid y)^2 \geq \exp(-H(\pi(\cdot \mid y))).$$

1177 Averaging both sides over Y and applying Jensen's inequality (since $x \mapsto e^{-x}$ is convex), we obtain:

$$1179 \mathbb{E}_{Z,Y}[\pi_n(Z)] = \mathbb{E}_Y \left[\sum_{z \in \mathcal{S}} \pi(z \mid Y)^2 \right] \\ 1180 \geq \mathbb{E}_Y \left[\exp(-H(\pi(\cdot \mid Y))) \right] \\ 1181 \geq \exp(-\mathbb{E}_Y[H(\pi(\cdot \mid Y))]) \\ 1182 = \exp(-H(Z \mid Y)) \\ 1183 = \exp(-H_0 + I(Z; Y)),$$

1186 where the last step uses the definition of mutual information.
 1187

Combining this with Equation (6) proves the claim. \square

Theorem C.5 shows that the success probability improves exponentially in the amount of mutual information gained through the lemmas, $I(Z; Y_{1:n})$. In particular, the success probability is strictly larger than in the case without lemmas, where $I(Z; Y_{1:n}) = 0$.

The following lemma was used in the proof of Theorem C.5:

Lemma C.6 (Relation Between Squared Sum and Entropy). *For any probability distribution $p = (p_i)_i$, the following inequality holds:*

$$\sum_i p_i^2 \geq \exp(-H(p)),$$

where $H(p) = -\sum_i p_i \log p_i$ denotes the Shannon entropy (with natural logarithm).

Proof. The log-sum inequality states that for nonnegative sequences $\{a_i\}, \{b_i\}$, the following holds:

$$\sum_i a_i \log \frac{a_i}{b_i} \geq \left(\sum_i a_i \right) \log \frac{\sum_i a_i}{\sum_i b_i}.$$

Let $a_i = p_i$ and $b_i = p_i^2$. Then the LHS becomes:

$$\sum_i p_i \log \frac{p_i}{p_i^2} = \sum_i p_i \log \frac{1}{p_i} = -\sum_i p_i \log p_i = H(p).$$

On the other hand, the right-hand side (RHS) becomes:

$$\left(\sum_i p_i \right) \log \frac{\sum_i p_i}{\sum_i p_i^2} = 1 \cdot \log \frac{1}{\sum_i p_i^2} = -\log \left(\sum_i p_i^2 \right).$$

Hence, the log-sum inequality gives:

$$H(p) \geq -\log \left(\sum_i p_i^2 \right).$$

Exponentiating both sides yields:

$$\sum_i p_i^2 \geq \exp(-H(p)).$$

This completes the proof. \square

D DETAILED EXPERIMENTAL SETUP

D.1 BENCHMARKING DATASET

We use the MiniF2F (Zheng et al., 2022) dataset, which consists of 488 mathematical problems formalized in Lean. These problems originate from sources such as AIME (American Invitational Mathematics Examination), AMC (American Mathematics Competitions), and IMO (International Math Olympiad) competitions, along with selected problems from the MATH dataset (Hendrycks et al., 2021), covering topics such as algebra, number theory, geometry, and analysis. Each problem is given as a Lean theorem statement. The benchmark is split into 244 validation and 244 test problems. We use the validation set during development (e.g., for tuning prompt formats) and report the final results on the test set. We use the revised version of miniF2F released by Wang et al. (2025); Ren et al. (2025).

Also, we observed that for problem names like `algebra_2varlineareq_fp3zeq11_3tfmlm5zeqn68_feqn10_zeq7`, the LLM often struggled to reliably reproduce the latter part of the name due to its unintelligible character sequence. Therefore, we modified such problem names by removing the less interpretable suffixes and replacing them with simpler, more memorable labels such as `algebra` for our experiments.

1242 D.2 USED MODELS
1243

1244 For the informal LLM, we use DeepSeek-R1-0528-Qwen3-8B⁵ (DeepSeek-AI, 2025), a
1245 model obtained by distilling the chain-of-thought outputs of DeepSeek-R1-0528 (DeepSeek-
1246 AI, 2025) into the Qwen3-8B (Yang et al., 2025a). This model surpasses Qwen3-8B
1247 on the AIME benchmark for natural language reasoning and achieves state-of-the-art per-
1248 formance at this scale. For the prover model, we use Goedel-Prover-V2-7B⁶ (Lin
1249 et al., 2025c) and DeepSeek-Prover-V2-7B⁷ (Ren et al., 2025), the state-of-the-art
1250 and second-best Lean 4 provers at this scale, respectively. For the formalizer model, we
1251 use Goedel-Formalizer-V2-8B⁸ (Lin et al., 2025c) in the Goedel-Prover setup and
1252 Kimina-Autoformalizer-7B⁹ (Wang et al., 2025). All of them are publicly available on
1253 Hugging Face (Wolf et al., 2020).

1254 D.3 IMPLEMENTATION DETAILS
1255

1256 All models are invoked via vLLM (Kwon et al., 2023), a high-performance inference engine for
1257 large language models. We set `max_num_batched_tokens` and `max_model_len` parameters
1258 to 16384 to accommodate the long context lengths required for theorem proving, while keeping all
1259 other settings at their vLLM defaults. The models are run on NVIDIA A100 GPUs with 40GB of
1260 memory. We use Lean version 4.9.0 (Moura & Ullrich, 2021) throughout all experiments, following
1261 the same setup in Xin et al. (2025a); Ren et al. (2025); Lin et al. (2025c).

1262 There are several bugs that may result in invalid Lean proofs being incorrectly accepted, such as
1263 the user-interference bug related to the `apply?` tactic discussed in version 2 of the arXiv paper
1264 by Ren et al. (2025), and a bug in REPL¹⁰. To avoid these issues and prevent invalid proofs from
1265 being mistakenly judged as correct, we check proofs with `lake build` instead of REPL and
1266 additionally verified that the `apply?` tactic is not used. Also, to avoid this bug and obtain reliable
1267 baseline results, we re-ran the experiments for Goedel-Prover-V2-8B. We used the official prompts
1268 provided on GitHub¹¹ and Hugging Face⁶, while keeping all other experimental settings strictly
1269 identical to those used in our method, thereby ensuring a fair comparison. For DeepSeek-Prover-
1270 V2, we relied on the results reported in version 2 of the arXiv paper (Ren et al., 2025), in which this
1271 bug has been fixed. All other baseline results are sourced from their respective papers.

1272 D.4 SAMPLE BUDGET
1273

1274 **MiniF2F.** We set $N_{\text{init}} = N_{\text{refine}} = 50$. Thus, the sample budget at the initial direct proving stage
1275 is 50 at the first iteration, and 100 in total when including iterative refinement. For lemmas, we use
1276 $N_{\text{init}} = N_{\text{refine}} = 10$ for each of the three lemmas. In the final synthesis stage, $N_{\text{init}} = N_{\text{refine}} = 50$
1277 is used again, resulting in a total sample budget of $50 + 50 + (10 + 10) \times 3 + 50 + 50 = 260$.

1278 **PutnamBench.** We set $N_{\text{init}} = N_{\text{refine}} = 20$. Thus, the sample budget at the initial direct proving
1279 stage is 20 at the first iteration, and 40 in total when including iterative refinement. For lemmas, we
1280 use $N_{\text{init}} = N_{\text{refine}} = 5$ for each of the three lemmas. In the final synthesis stage, $N_{\text{init}} = N_{\text{refine}} =$
1281 20 is used again, resulting in a total sample budget of $20 + 20 + (5 + 5) \times 3 + 20 + 20 = 110$.

1283 D.5 BASELINE METHODS
1284

1285 We compare our approach against several baseline methods, categorized into two main classes:
1286 tree search methods and whole-proof generation methods. Tree search methods construct proofs
1287 incrementally by predicting individual tactics step by step, often guided by search algorithms such as
1288 best-first search or Monte Carlo Tree Search (MCTS). In contrast, whole-proof generation methods

1289 ⁵<https://huggingface.co/deepseek-ai/DeepSeek-R1-0528-Qwen3-8B>

1290 ⁶<https://huggingface.co/Goedel-LM/Goedel-Prover-V2-8B>

1291 ⁷<https://huggingface.co/deepseek-ai/DeepSeek-Prover-V2-7B>

1292 ⁸<https://huggingface.co/Goedel-LM/Goedel-Formalizer-V2-8B>

1293 ⁹<https://huggingface.co/AI-MO/Kimina-Autoformalizer-7B>

1294 ¹⁰<https://github.com/leanprover-community/repl/issues/44>

1295 ¹¹<https://github.com/Goedel-LM/Goedel-Prover-V2>

1296 attempt to generate an entire proof script in a single forward pass, relying on the model’s ability to
 1297 plan the proof holistically.
 1298

1299 The overview of the baseline methods used in our experiments is as follows:
 1300

1301 **Tree Search Method:**

- 1302 • **DeepSeek-Prover-V1.5-RL + RMaxTS** (Xin et al., 2025a) uses DeepSeek-Prover-V1.5-
 1303 RL (Xin et al., 2025a), a 7B model trained with reinforcement learning, combined with
 1304 RMaxTS (Xin et al., 2025a), a variant of MCTS that uses intrinsic rewards to explore
 1305 diverse proof paths.
 1306
- 1307 • **InternLM2.5-StepProver-BF + CG** (Wu et al., 2024a) uses InternLM2.5-StepProver (Wu
 1308 et al., 2024a), a 7B model trained via expert iteration (Anthony et al., 2017; Polu et al.,
 1309 2023) starting with InternLM2-StepProver (Wu et al., 2024b), combined with a best-first
 1310 search (BFS) strategy and a critic-guided (CG) sampling technique to explore longer proofs
 1311 effectively.
 1312
- 1313 • **HunyuanProver v1.6 + BFS + DC** (Li et al., 2025) uses HunyuanProver, a 7B model fine-
 1314 tuned via a scalable data synthesis pipeline, in conjunction with best-first search guided by
 1315 the distance critic (DC) to efficiently navigate complex Lean 4 proof search spaces.
 1316
- 1317 • **BFS-Prover** (Xin et al., 2025b) uses a fine-tuned model of Qwen2.5-Math-7B model (Yang
 1318 et al., 2024), trained through an expert-iteration pipeline. During inference, it employs a
 1319 best-first search strategy to navigate the proof space efficiently.
 1320

1321 **Whole-Proof Generation Methods:**
 1322

- 1323 • **Leanabell-Prover-GD-RL** (Zhang et al., 2025) is a 7B model post-trained through con-
 1324 tinual training on statement-proof pairs and reinforcement learning using Lean 4 outcome
 1325 rewards. This model is a fine-tuned version of Goedel-Prover-SFT (Lin et al., 2025b).
 1326
- 1327 • **Goedel-Prover-SFT** (Lin et al., 2025b) is a 7B-parameter model obtained by supervised
 1328 fine-tuning on DeepSeek-Prover-V1.5-Base (Xin et al., 2025a) with expert-iteration.
 1329
- 1330 • **STP: Self-Play Theorem Prover** (Dong & Ma, 2025) employs a self-play framework that
 1331 simultaneously takes on two roles, conjecturer and prover. The conjecturer is iteratively
 1332 trained on statements that are barely provable by the current prover, incentivizing it to
 1333 generate increasingly challenging conjectures. The prover uses standard expert iteration to
 1334 verify and prove the generated conjectures. This model is a fine-tuned version of DeepSeek-
 1335 Prover-V1.5-SFT (Xin et al., 2025a), which is a 7B-parameter model.
 1336
- 1337 • **Kimina-Prover-Preview** (Wang et al., 2025) is a 72B-parameter reasoning model that
 1338 learns specialized formal reasoning patterns via reinforcement learning. It is pretrained on
 1339 a large corpus of formal proofs and fine-tuned with a binary correctness reward and con-
 1340 sistency penalty. They also provide **Kimina-Prover-Preview-Distill-7B**, a distilled version
 1341 from the 72B model.
 1342
- 1343 • **DeepSeek-Prover-V2** (Ren et al., 2025) uses DeepSeek-V3 to decompose each theorem
 1344 into subgoals and then employs the proofs of those subgoals as cold-start data for rein-
 1345 forcement learning using binary correctness rewards and a consistency penalty to ensure
 1346 that every subgoal appears in the final proof. It is implemented as a 671B-parameter model,
 1347 and a distilled 7B-parameter variant is also provided.
 1348
- 1349 • **Leanabell-Prover-V2** (Ji et al., 2025) is a 7B-parameter prover obtained by post-training
 1350 existing models with verifier-integrated reinforcement learning. Two variants are pro-
 1351 vided: **Leanabell-Prover-V2-KM**, which is post-trained from Kimina-Prover-Preview-
 1352 Distill-7B (Wang et al., 2025), and **Leanabell-Prover-V2-DS**, which is post-trained from
 1353 DeepSeek-Prover-V2-7B (Ren et al., 2025).
 1354
- 1355 • **Goedel-Prover-V2** (Lin et al., 2025c) is a series of open-source provers built on expert-
 1356 iteration and reinforcement learning, augmented with (i) scaffolded data synthesis (curric-
 1357 ula of increasingly difficult synthetic theorems), (ii) verifier-guided self-correction using
 1358 Lean feedback, and (iii) model averaging.
 1359

1350
1351

D.6 COMPARISON IN TERMS OF TOTAL TOKEN BUDGET

1352
1353
1354
1355
1356
1357
1358

In our pipeline, the informal LLM is used only in three places: (i) Initial direct proving without iterative refinement, which is invoked 50 times (once for each generation), (ii) Lemma generation, which is invoked once, and (iii) Initial direct proving for each generated lemma without iterative refinement, which is invoked 10 times for each of the three lemmas. The formalizer model is used only three times to formalize the three generated lemmas. Outside of these calls, the pipeline does not invoke any additional LLMs; the remaining stages only execute Lean or reuse already proved lemmas without consuming new tokens.

1359
1360
1361
1362
1363
1364

Thus, in addition to the 260 prover calls reported in Table 1, Prover Agent uses only $50 + 1 + 3 \times 10 + 3 = 84$ extra LLM calls, resulting in a total of $260 + 84 = 344$ LLM executions. Because the context length is fixed for all calls, the total token budget is effectively proportional to this number of LLM invocations. Also, when informal proofs, Lean feedback, or proved lemmas occupy part of the prompt, the corresponding output token length simply decreases, since the context size of the model is predefined. Thus, the total token consumption is governed by the number of LLM calls.

1365
1366
1367
1368
1369
1370

Importantly, with this total token budget corresponding to 344 LLM calls, Prover Agent achieves: 88.1% in the ensemble setting, 86.5% in the GoedelProver-V2 setting, and 82.8% in the DeepSeek-Prover-V2 setting. These results surpass the corresponding baseline performance of GoedelProver-V2, which uses 512 LLM calls, as well as the corresponding baselines of DeepSeek-Prover-V2, which use 1,024 and 8,192 LLM calls. Therefore, even when measured in total token budget, Prover Agent achieves a higher success rate using fewer tokens than the corresponding baselines.

1371
1372
1373
1374

E EXAMPLES OF SUCCESSFUL CASES ENABLED BY LEMMAS AND ITERATIVE REFINEMENT

1375
1376
1377

In Appendices E.1 and E.2, we present and analyze an example successfully solved via a lemma and an example successfully solved through iterative refinement, respectively.

1378
1379
1380
1381
1382
1383
1384
1385

E.1 CASE STUDY OF SUCCESSFUL EXAMPLE WITH LEMMAS

E.1.1 DETAILED ANALYSIS

We analyze in detail the reasoning process for the problem `induction_nf_fact_lt_exp_nm1ngt3`, a case where the direct proof attempt failed but the use of auxiliary lemmas led to a successful proof. This problem asks for a formal proof that, for all natural numbers $n > 3$, the inequality $n! < n^{n-1}$ always holds.

1386
1387
1388
1389
1390
1391
1392
1393
1394
1395

The outputs for this problem, such as the generated lemmas, final formal proof, and the associated reasoning process, are provided in Appendix E.1.2 and after.

In this case, the agent generated the following three lemmas: The first states that $3! < 3^{3-1}$; the second states that for any natural number $n \geq 2$, $n^{n-1} < (n+1)^{n-1}$; and the third states that for any natural number $n \geq 3$, $n! < (n+1)^{n-1}$. The first is a special case of the original problem with $n = 3$, while the second may provide a helpful hint toward solving the original problem. Both were easily proven in a single direct proof attempt. The third lemma generated in this case asserts that for any natural number $n \geq 3$, $n! < (n+1)^{n-1}$. This lemma closely resembles the original problem, as it is a slightly weaker version of its conclusion. Due to its similarity and retained difficulty, the agent failed to construct a direct proof for it.

1396
1397
1398
1399
1400
1401
1402
1403

By examining the final successful reasoning trace in Appendix E.1.5, we see that the special case for $n = 3$, considered as the first lemma, appears explicitly on line 7. The reasoning also checks the cases for $n = 4$ and $n = 5$, following a similar pattern. Furthermore, as stated on line 13, the use of mathematical induction is clearly identified as the intended proof strategy. Then, the reasoning trace from line 14 to line 80 further elaborates the proof process within the framework of mathematical induction. Furthermore, in the final proof, the proof technique used in Lemma 2 is explicitly applied at lines 195–196.

Next, as a comparison, we analyze the reasoning process from the initial direct proving attempt without using any lemmas, as shown in Appendix E.1.6. Here, we present the reasoning trace that

1404 resulted in the fewest Lean errors among all initial direct attempts. Compared to the successful case
 1405 with lemmas, we see that the proof strategy is much less clear in this direct attempt. In the “Key
 1406 Observations” section (lines 6 to 14), there is no indication of using mathematical induction, unlike
 1407 in the lemma-assisted case. Although the system explores several ideas from lines 15 to 63, the
 1408 reasoning appears less focused and more exploratory, lacking a concrete plan. As a result, while it
 1409 eventually leans toward using induction, the lack of a clear and structured approach prevents it from
 1410 working out the necessary details, ultimately leading to failure in the formal proof, which tolerates
 1411 no ambiguity.

1412 This detailed case study highlights the effectiveness of our lemma-generation approach in uncov-
 1413 ering viable proof strategies. This marks a significant advance over prior methods that decompose
 1414 problems into subgoals, which often assume the overall proof strategy is known in advance. Identifi-
 1415 cating an initial proof strategy is often a challenging part of solving difficult problems. Indeed, Ren
 1416 et al. (2025) employs a decomposition-based approach but relies on the much larger and stronger
 1417 DeepSeek-V3 (DeepSeek-AI, 2024) to formulate the initial proof sketch. In contrast, our agent fol-
 1418 lows a reasoning process similar to that of human mathematicians when the proof strategy is not
 1419 apparent at first glance, exploring special cases or hypothesizing intermediate steps to discover a
 1420 promising direction and ultimately uncover the overall proof strategy.

1421 E.1.2 LEAN ENVIRONMENT SETUP

1423 All Lean code was executed with the following header, following Xin et al. (2025a); Ren et al.
 1424 (2025), which is omitted in the examples for brevity:

```
1426   1 import Mathlib
1427   2 import Aesop
1428   3
1429   4 set_option maxHeartbeats 0
1430   5
1431   6 open BigOperators Real Nat Topology Rat
```

1432 Note that the following Lean code works correctly with Lean version 4.9.0, the environment used
 1433 in our experiments, which strictly follows the same setup in Xin et al. (2025a); Ren et al. (2025).
 1434 However, `Nat.pow_lt_pow_of_lt_left` has been deprecated in newer versions of Lean, so
 1435 it must be replaced with `Nat.pow_lt_pow_left` to work with the latest environment.

1436 E.1.3 THE PROBLEM

1438 The following problem asks for a formal proof that, for all natural numbers n greater than 3, the
 1439 inequality $n! < n^{n-1}$ holds:

```
1441   1 theorem induction_nfactltexpnm1ngt3
1442   2   (n : ℕ)
1443   3   (h0 : 3 ≤ n) :
1444   4   (n)! < n^(n - 1) := by sorry
```

1446 E.1.4 THE GENERATED LEMMAS

1448 The generated lemmas are shown below, each presented with both its original natural language form
 1449 and the corresponding Lean formalization. The first and second lemmas were successfully proved
 1450 and are presented with their proofs. The third lemma could not be proved and is therefore presented
 1451 with a `sorry` placeholder.

```
1452   1 ### Lemma 1: base_case_3
1453   2 **Assumptions**:
1454   3 None
1455   4
1456   5 **Conclusion**:
1457   6 3! < 3^(3-1)
```

```

1458
1459 1 theorem base_case_3 : (3 : ℕ)! < 3^(3 - 1) := by
1460 2 have h_main : (3 : ℕ)! < 3^(3 - 1) := by
1461 3 -- Calculate the factorial and the power step-by-step
1462 4 norm_num [Nat.factorial, Nat.pow_succ, Nat.mul_assoc]
1463 5 -- Use 'decide' to confirm the inequality
1464 6 < ; > decide
1465 7 exact h_main
1466
1467 1 ### Lemma 2: exponent_inequality
1468 2 **Assumptions**:
1469 3 n is a natural number and n ≥ 2
1470 4
1471 5 **Conclusion**:
1472 6 n^(n-1) < (n+1)^(n-1)
1473
1474 1 theorem exponent_inequality {n : ℕ} (hn : 2 ≤ n) :
1475 2 n^(n - 1) < (n + 1)^(n - 1) := by
1476 3 have h_main : n^(n - 1) < (n + 1)^(n - 1) := by
1477 4 have h0 : n - 1 + 1 = n := by
1478 5 omega
1479 6 have h1 : n ≥ 2 := by omega
1480 7 have h2 : n - 1 ≥ 1 := by omega
1481 8 have h3 : n < n + 1 := by omega
1482 9 -- Use the fact that if a < b and n > 0, then a^n < b^n
1483 10 exact calc
1484 11 n^(n - 1) < (n + 1)^(n - 1) := by
1485 12 -- Apply the lemma that if a < b and n > 0, then a^n < b^n
1486 13 exact Nat.pow_lt_pow_of_lt_left h3 (by omega)
1487 14 _ = (n + 1)^(n - 1) := by rfl
1488 15 exact h_main
1489
1490 1 ### Lemma 3: factorial_less_than_n_next_power
1491 2 **Assumptions**:
1492 3 n is a natural number and n ≥ 3
1493 4
1494 5 **Conclusion**:
1495 6 n! < (n+1)^(n-1)
1496
1497 1 theorem factorial_lt_n_next_power {n : ℕ} (hn : 3 ≤ n) :
1498 2 n ! < (n + 1)^(n - 1) := by sorry
1499
1500
1501 1 ### Detailed Proof and Analysis
1502 2
1503 3 We are tasked with proving that for all natural numbers 'n ≥ 3', the factorial of 'n', i.e., 'n!', is less than 'n^(n - 1)'.
1504 4
1505 5 #### Key Observations:
1506 6 1. **Base Cases**:
1507 7 - For 'n = 3': '3! = 6' and '3^(3 - 1) = 3^2 = 9'. So, '6 < 9' holds.
1508 8 - For 'n = 4': '4! = 24' and '4^(4 - 1) = 4^3 = 64'. So, '24 < 64' holds.
1509 9 - For 'n = 5': '5! = 120' and '5^(5 - 1) = 5^4 = 625'. So, '120 < 625' holds.
1510
1511

```

```

1512      - The pattern suggests that this holds for all ' $n \geq 3$ '.
1513
1514 12 2. General Proof Idea**:
1515      - The proof can proceed by mathematical induction. We first
1516      check the base case ' $n = 3$ ', and then for the inductive step, we
1517      assume the statement holds for ' $n = k$ ' (i.e., ' $k! < k^{(k-1)}$ '')
1518      and show that it also holds for ' $n = k + 1$ ' (i.e., ' $(k+1)! < (k +$ 
1519       $1)^{k+1}$ ').
1520      - The inductive step involves showing that ' $(k+1)! = (k+1) *$ 
1521       $k! < (k+1) * k^{(k-1)}$ '. The last expression is ' $(k+1)^k$ ' because ' $k^{(k-1)} = k^k / k$ ' when multiplying by ' $(k+1)$ ' is not
1522      clean. Instead, we can directly use the fact that ' $(k+1)! = (k +$ 
1523       $1) * k!$ ' and ' $k! < k^{(k-1)}$ ' by the inductive hypothesis, but we
1524      must also ensure ' $k+1 < k+1+1$ ' is not relevant here. Wait,
1525      perhaps the best approach is to directly manipulate the inequality
1526      ' $(k+1)! < (k+1)^k$ '.
1527
1528 16      However, this seems messy. A simpler way is to compare ' $(k +$ 
1529       $1)! < (k+1)^k$ ' directly. For ' $k \geq 3$ ', we have ' $(k+1)! = (k$ 
1530       $+ 1) * k! < (k+1) * k^k$ '.
1531
1532 17      But this is not obviously true because ' $k! < k^k$ ' is equivalent
1533      to ' $k < k+1$ ', which is trivially true.
1534
1535 18      Alternatively, ' $(k+1)! = (k+1) * k! < (k+1) * k^{(k-1)}$ ' by
1536      IH, and ' $(k+1) * k^{(k-1)} = k^k + k^{(k-1)}$ ', which is less
1537      than ' $(k+1)^k = k^k + k * k^{(k-1)} + \dots$ ', but this is getting
1538      complicated.
1539
1540 21      A much better approach is to use induction directly. For ' $n =$ 
1541      3', we already checked this. Assume ' $k! < k^{(k-1)}$ ' for some ' $k \geq$ 
1542      3'. Then:
1543      ' $(k+1)! = (k+1) * k! < (k+1) * k^{(k-1)}$ '. Now, ' $(k+1) *$ 
1544       $k^{(k-1)} = k^k + k^{(k-1)}$ '. We need this to be ' $< (k+1)^k =$ 
1545       $k^k + k * k^{(k-1)} + \dots$ '.
1546
1547 24      But ' $k^k$ ' cancels out, and we need ' $k^{(k-1)} < k * k^{(k-1)} +$ 
1548       $\dots$ ', i.e., ' $1 < k + \dots$ ', which is true for ' $k \geq 3$ '.
1549
1550 27      However, the last step is not straightforward, and it might be
1551      simpler to use induction concretely.
1552
1553 29 ##### Step 1: Abstract Plan
1554
1555 31 1. Base Case ('n = 3')**:
1556      - Compute ' $3! = 6$ ' and ' $3^{(3-1)} = 9$ '. Since ' $6 < 9$ ', the
1557      base case holds.
1558
1559 34 2. Inductive Step ('n = k + 1')**:
1560      - Assume ' $k! < k^{(k-1)}$ ' holds for some ' $k \geq 3$ '.
1561      - We need to show ' $(k+1)! < (k+1)^k$ '.
1562      - ' $(k+1)! = (k+1) * k! < (k+1) * k^{(k-1)}$ ' by the
1563      inductive hypothesis.
1564      - Next, we need to compare ' $(k+1) * k^{(k-1)}$ ' to ' $(k+1)^k =$ 
1565       $(k+1) * k^k / k$ '.
1566      - The inequality ' $k * (k+1) \leq k * k + k * k = 2 * k^2$ ' for ' $k$ 
1567       $\geq 1$ ' is too loose. A better approach is as follows:
1568      - We want ' $(k+1) * k^{(k-1)} < (k+1)^k$ '.
1569      - Divide both sides by ' $(k+1)$ ': ' $k^{(k-1)} < (k+1)^{(k-1)}$ '.
1570      - But:
1571      - This is false for ' $k = 3$ ': ' $3^2 = 9 < 4^2 = 16$ ' holds.
1572      For ' $k = 4$ ': ' $4^3 = 64 < 5^3 = 125$ ' holds, etc. But we need ' $(k +$ 
1573       $1) * k^{(k-1)} < (k+1)^k$ ', not ' $k^{(k-1)} < (k+1)^{(k-1)}$ '.
1574      - Alternatively, perhaps a direct proof is better:

```

```

1566      -  $(k + 1) * k^{(k - 1)} = k^k + k^{(k - 1)}$ .
1567      - We must show  $k^k + k^{(k - 1)} < (k + 1)^k = k^k + k * k^{(k - 1)} + \dots + 1$ .
1568      - Since  $k \geq 3$ ,  $k^{(k - 1)} < k * k^{(k - 1)}$  and  $0 < k^k$  (which cancels), we get  $k^{(k - 1)} < k * k^{(k - 1)}$  because  $1 < k$  when  $k \geq 2$ , so  $k^k + k^{(k - 1)} < k^k + k * k^{(k - 1)} + \dots$  is justified.
1569
1570
1571
1572
1573      But the above is still messy. A better way is to prove  $(k + 1) * k^{(k - 1)} < (k + 1)^k$  for  $k \geq 3$ :
1574
1575       $(k + 1) * k^{(k - 1)} = k^k + k^{(k - 1)}$ , and  $(k + 1)^k = k^k + k * k^{(k - 1)} + \dots$ .
1576      Subtract:
1577       $(k + 1)^k - (k + 1) * k^{(k - 1)} = k^k + k * k^{(k - 1)} + \dots - (k^k + k^{(k - 1)}) = k * k^{(k - 1)} + \dots$  which is clearly positive because all terms are positive.
1578
1579      Alternatively, just note that  $k^k > 0$  when  $k \geq 1$  is sufficient because  $(k + 1)^k = k^k + \dots \geq k^k > k^{(k - 1)}$ .
1580
1581
1582      However, the direct approach is to compute  $(k + 1)^k = (k + 1) * ((k + 1) * \dots * 1) \geq (k + 1) * (k * \dots * 1) = (k + 1) * k^{(k - 1)}$ , which would work except for the last factor.
1583
1584
1585      Wait, we need  $(k + 1)^k > (k + 1) * k^{(k - 1)}$ . But  $(k + 1)^k = (k + 1) * (k + 1)^{(k - 1)}$ .
1586
1587
1588      But  $(k + 1)^{(k - 1)} \geq k^{(k - 1)}$  is not immediately obvious (and may even be false for  $k = 3$  ( $4^2 = 16 \geq 8 = 3^2$  is true, but  $4^3 = 64 \geq 3^3 = 27$  is true)).
1589
1590
1591      This seems correct, but we instead keep it simple:
1592
1593      ##### Inductive Proof Sketch:
1594      1. Base Case ('n = 3'): '6 < 9' is true.
1595      2. Inductive Step ('n = k + 1' assuming  $k! < k^{(k - 1)}$ ):
1596          - We have  $(k + 1)! < (k + 1) * k^{(k - 1)}$ .
1597          - We need  $(k + 1) * k^{(k - 1)} \leq (k + 1)^k = (k + 1) * k^{(k - 1)} + \dots$ 
1598          - Alternatively, the most straightforward induction is simply to use that  $(k + 1) * k^{(k - 1)} < (k + 1)^k$  when  $k \geq 3$ .
1599
1600
1601      This can be proved as follows:
1602       $(k + 1)^k = (k + 1) * (k + 1)^{(k - 1)}$  and  $(k + 1)^{(k - 1)} \geq k^{(k - 1)}$  for  $k \geq 3$  because:
1603       $(k + 1)^2 = k^2 + 2k + 1 \geq k^2$  for all  $k \geq 1$ ,
1604       $(k + 1)^3 = k^3 + 3k^2 + 3k + 1 \geq k^3 + 3k^2 \geq k^3 + 2k^2 \geq k^3$  for  $k \geq 2$ , etc.
1605
1606      But we get  $(k + 1)^k \geq (k + 1) * k^{(k - 1)}$ , which is not what we want! This is because we made a miscount. The correct relation is:
1607       $(k + 1)^k = (k + 1) * (k + 1)^{(k - 1)}$ , and  $(k + 1)^{(k - 1)} > k^{(k - 1)}$  for  $k \geq 3$  (since  $l < m$  implies  $l^{(n)} < m^{(n)}$  for  $l, m > 1$ ). That is,  $(k + 1)^k > (k + 1) * k^{(k - 1)} > (k + 1) * k^{(k - 1)} - k^{(k - 1)}$  is not true.
1608
1609
1610
1611      Instead,  $(k + 1)^k = (k + 1) * (k + 1)^{(k - 1)} \geq (k + 1) * k^{(k - 1)}$  because  $(k + 1)^{(k - 1)} \geq k^{(k - 1)}$  for  $k \geq 3$ .
1612
1613
1614      Therefore, the full chain is:
1615       $(k + 1)! = (k + 1) * k! < (k + 1) * k^{(k - 1)} \leq (k + 1)^k$ .
1616
1617      Thus, we can prove the inductive step using the above reasoning.
1618
1619      ##### Abstract Plan:
1620      1. Base Case:

```

```

1620     - Verify the inequality for 'n = 3'.
1621 2. Inductive Step:
1622     - Assume the inequality holds for 'n = k', i.e., ' $k! < k^{(k-1)}$ '.
1623     - for 'k ≥ 3'.
1624     - Prove for 'n = k + 1', i.e., ' $(k+1)! < (k+1)^k = (k+1) * k^{(k-1)}$ '.
1625     - But it is enough to prove ' $(k+1)^k ≥ (k+1) * k^{(k-1)}$ ':
1626     - Since ' $(k+1)^k = (k+1) * (k+1)^{k-1}$ ', we must show
1627     ' $(k+1)^{k-1} ≥ k^{(k-1)}$ '.
1628     - This is true for 'k ≥ 4' since ' $(k+1)/k ≥ k/(k-1)$ '.
1629     - Check 'k = 3' separately: ' $4^2 = 16 ≥ 3^2 = 9$ ' and ' $4^3 = 64 ≥ 3^3 = 27$ ', etc.
1630
1631 95 ##### Lean 4 Proof Sketch with 'have' Statements
1632
1633 97 ````lean4
1634 98 theorem induction
1635   (n :  $\mathbb{N}$ )
1636   (h0 :  $3 \leq n$ ) :
1637   (n)! < n(n-1) := by
1638   -- Base case: n = 3
1639   have h_base : (3 :  $\mathbb{N}$ )! < 3(3-1) := by sorry
1640   -- Inductive step: assume  $(k)! < k^{(k-1)}$  for some  $k \geq 3$ , prove
1641   (k+1)! < (k+1)((k+1)-1)
1642   have h_inductive_step :  $\forall (k : \mathbb{N})$ ,  $3 \leq k \rightarrow (k)! < k^{(k-1)} \rightarrow$ 
1643   (k+1)! < (k+1)((k+1)-1) := by sorry
1644   -- Combine base case and inductive step to get the general result
1645   have h_general : (n)! < n(n-1) := by sorry
1646   sorry
1647   ````

1648 111 This sets up the scaffolding of the proof using 'have' statements
1649 with 'sorry'.
1650 112 ##### Step-by-Step Abstract Plan
1651 113 1. Base Case ('n = 3'):
1652   - Directly check ' $3! = 6$ ' and ' $3^{(3-1)} = 9$ ', hence ' $6 < 9$ '.
1653 114 2. Inductive Hypothesis ('n = k'):
1654   - Assume ' $k! < k^{(k-1)}$ ' for some ' $k \geq 3$ '.
1655 115 3. Inductive Step ('n = k + 1'):
1656   - Prove ' $(k+1)! < (k+1)^{((k+1)-1)}$ ' using:
1657   - ' $(k+1)! = (k+1) * k! < (k+1) * k^{(k-1)}$ ' by IH.
1658   - ' $(k+1)^k = (k+1) * (k+1)^{k-1} \geq (k+1) * k^{(k-1)}$ ', if ' $(k+1)^{k-1} \geq k^{(k-1)}$ '. This holds for 'k ≥ 3'
1659   because ' $(k+1) \geq k$ ' and functions are increasing.
1660
1661 116 4. Conclusion:
1662   - By induction, the statement holds for all ' $n \geq 3$ '.
1663
1664 129 ##### Lean 4 Proof Sketch with 'have' Statements and 'sorry's
1665
1666 131 ````lean4
1667 132 theorem induction
1668   (n :  $\mathbb{N}$ )
1669   (h0 :  $3 \leq n$ ) :
1670   (n)! < n(n-1) := by
1671   have h_base : (3 :  $\mathbb{N}$ )! < 3(3-1) := by sorry
1672   have h_inductive_step :  $\forall (k : \mathbb{N})$ ,  $3 \leq k \rightarrow (k)! < k^{(k-1)} \rightarrow$ 
1673   (k+1)! < (k+1)((k+1)-1) := by sorry
1674   have h_general : (n)! < n(n-1) := by sorry
1675   sorry
1676   ````
```

```

1674
1675 141  (Note: The `have` statements are used to sketch the high-level
1676 142  structure of the proof. The actual Lean 4 proof would fill in the
1677 143  `sorry` placeholders with detailed reasoning.)
1678 144  ### Complete Lean 4 Proof
1679 145
1680 146  ````lean4
1681 147  theorem induction
1682 148  (n :  $\mathbb{N}$ )
1683 149  (h0 :  $3 \leq n$ ) :
1684 150  (n)! <  $n^{\wedge}(n - 1)$  := by
1685 151  have hbase : (3 :  $\mathbb{N}$ )! <  $3^{\wedge}(3 - 1)$  := by
1686 152  norm_num
1687 153  <;> decide
1688 154
1689 155  have hinductive_step :  $\forall (k : \mathbb{N}), 3 \leq k \rightarrow (k)! < k^{\wedge}(k - 1) \rightarrow$ 
1690 156   $(k + 1)! < (k + 1)^{\wedge}((k + 1) - 1)$  := by
1691 157  intro k hk' hk''
1692 158  have h1 :  $(k + 1)! = (k + 1) * k!$  := by
1693 159  simp [Nat.factorial_succ, Nat.mul_comm]
1694 160  <;> ring
1695 161  <;> simp_all [Nat.factorial]
1696 162  <;> ring
1697 163  <;> simp_all [Nat.factorial]
1698 164  rw [h1]
1699 165  have h2 :  $(k + 1 : \mathbb{N}) * k! < (k + 1 : \mathbb{N}) * k^{\wedge}(k - 1)$  := by
1700 166  -- Apply the inductive hypothesis to get the inequality for
1701 167  'k! < k^(k - 1)'
1702 168  have h3 :  $k! < k^{\wedge}(k - 1)$  := hk''
1703 169  have h4 :  $(k + 1 : \mathbb{N}) > 0$  := by linarith
1704 170  have h5 :  $(k + 1 : \mathbb{N}) * k! < (k + 1 : \mathbb{N}) * k^{\wedge}(k - 1)$  := by
1705 171  exact Nat.mul_lt_mul_of_pos_left h3 (by linarith)
1706 172  exact h5
1707 173  have h3 :  $(k + 1 : \mathbb{N})^{\wedge}((k + 1) - 1) = (k + 1)^{\wedge}k$  := by
1708 174  simp [Nat.sub_add_cancel (by linarith :  $1 \leq k + 1$ )]
1709 175  <;> ring_nf
1710 176  <;> simp_all
1711 177  rw [h3]
1712 178  -- Use the fact that ' $(k + 1) * k^{\wedge}(k - 1) < (k + 1)^{\wedge}k$ ' to
1713 179  conclude the proof
1714 180  have h4 :  $(k + 1 : \mathbb{N}) * k^{\wedge}(k - 1) < (k + 1)^{\wedge}k$  := by
1715 181  have h5 :  $k \geq 3$  := by exact_mod_cast hk'
1716 182  have h6 :  $(k + 1 : \mathbb{N}) * k^{\wedge}(k - 1) < (k + 1)^{\wedge}k$  := by
1717 183  have h7 :  $(k + 1 : \mathbb{N})^{\wedge}k = (k + 1) * (k + 1)^{\wedge}(k - 1)$  := by
1718 184  cases k with
1719 185  | zero => contradiction
1720 186  | succ k' =>
1721 187  cases k' with
1722 188  | zero => contradiction
1723 189  | succ k'' =>
1724 190  cases k'' with
1725 191  | zero => contradiction
1726 192  | succ k''' =>
1727 193  simp [Nat.pow_succ, Nat.mul_assoc]
1728 194  <;> ring_nf at *
1729 195  <;> omega
1730 196  rw [h7]
1731 197  have h8 :  $(k + 1 : \mathbb{N}) * k^{\wedge}(k - 1) < (k + 1) * ((k + 1)^{\wedge}(k - 1))$  := by
1732 198  have h9 :  $k^{\wedge}(k - 1) < (k + 1)^{\wedge}(k - 1)$  := by
1733 199  exact Nat.pow_lt_pow_of_lt_left (by linarith) (by
1734 200  have h10 :  $k - 1 \geq 2$  := by
1735 201  omega

```

```

1728           omega)
1729           nlinarith
1730           nlinarith
1731           exact h6
1732           nlinarith
1733
1734   have h_general : (n)! < n ^ (n - 1) := by
1735     have h1 : ∀ n : ℕ, 3 ≤ n → (n)! < n ^ (n - 1) := by
1736       intro n hn
1737       induction' hn with k hk IH
1738       . -- Base case: k = 3
1739         simp [Nat.factorial]
1740         <;> norm_num
1741       . -- Inductive step: assume the statement holds for k, prove
1742       for k + 1
1743         cases k with
1744           | zero => contradiction
1745           | succ k' =>
1746             cases k' with
1747               | zero => contradiction
1748               | succ k'' =>
1749                 cases k'' with
1750                   | zero => contradiction
1751                   | succ k''' =>
1752                     simp_all [Nat.factorial, Nat.pow_succ, Nat.mul_assoc]
1753                     -- Use the inductive hypothesis and the lemma
1754                     'h_inductive_step' to prove the next step
1755                     <;>
1756                     (try omega) <;>
1757                     (try nlinarith) <;>
1758                     (try
1759                       {
1760                         have h2 := h_inductive_step (k''' + 3) (by
1761                           linarith) (by simpa [Nat.factorial, Nat.pow_succ, Nat.mul_assoc]
1762                           using IH)
1763                         simp_all [Nat.factorial, Nat.pow_succ,
1764                           Nat.mul_assoc]
1765                         <;>
1766                         (try
1767                           {
1768                             aesop
1769                           })
1770                         <;>
1771                         (try
1772                           {
1773                             nlinarith [pow_pos (by norm_num : (0 : ℕ) < 2)
1774                               k''', pow_pos (by norm_num : (0 : ℕ) < 2) (k''' + 1), pow_pos (by
1775                               norm_num : (0 : ℕ) < 2) (k''' + 2)]
1776                           })
1777                         exact h1 n h0
1778
1779           exact h_general
1780           ```
1781

```

E.1.6 COMPARISON: DIRECT PROOF ATTEMPT WITHOUT LEMMAS

For comparison, we present below the reasoning process and generated Lean code from a direct proof attempt without using auxiliary lemmas. Among the 100 initial direct proof attempts ($N_{\text{init}} = 100$), we show the one that resulted in the fewest Lean errors.

```

1782
1783     1  ### Detailed Proof and Analysis
1784
1785     2
1786     3  **Problem Analysis:***
1787     4  We need to prove that for all integers ' $n \geq 3$ ', ' $n! < n^{(n-1)}$ '.
1788
1789     5
1790     6  **Key Observations:***
1791     7  1. For ' $n \geq 3$ ', ' $n-1 \geq 2$ ' (since ' $3-1=2$ '), so ' $n^{(n-1)}$ ' grows faster than ' $n!$ ' as ' $n$ ' increases.
1792     8  2. The factorial ' $n!$ ' is the product of all integers from ' $1$ ' to ' $n$ ', while ' $n^{(n-1)}$ ' is a product of ' $n-1$ ' copies of ' $n$ ' (assuming ' $n \geq 3$ ').
1793     9  3. The inequality ' $n! < n^{(n-1)}$ ' can be compared term by term, but a more precise estimate is needed.
1794    10  4. For ' $n \geq 3$ ', we have ' $n! < (n+1-1)! = n!$ '? Wait, this is not helpful. Instead, we can compare ' $n!$ ' with ' $n^{(n-1)}$ ':
1795    11      - The inequality ' $n! < n^{(n-1)}$ ' is equivalent to ' $(n-1)! < n^{(n-2)}$ ', but this is not directly useful either.
1796    12  5. A better idea is to directly compare ' $n!$ ' with ' $n*(n-1)*...*(2)$ ' and ' $n^{(n-1)} = n * n * ... * n$ '.
1797    13      - For ' $n \geq 3$ ', each term ' $k$ ' in the product ' $2 * 3 * ... * (n-1)$ ' is ' $\geq 1$ ' (except ' $2 * 1$ ' is skipped if ' $n \geq 4$ ', but ' $n \geq 3$ ' is given), and each is ' $\leq n-1$ ' for ' $k \leq n-2$ '. However, comparing ' $k$ ' to ' $n$ ' is complicated.
1798    14      - Alternatively, ' $n!$ ' is the product of ' $n$ ' factors, each ' $\leq n$ ', but ' $n!$ ' is a product of all numbers from ' $1$ ' to ' $n$ ', whereas ' $n^{(n-1)}$ ' is a product of ' $n-1$ ' copies of ' $n$ '.
1799
1800    15  16 But actually, induction is not straightforward here. The key is to compare ' $n!$ ' with ' $n * (n-1)^{(n-1)}$ ', since ' $(n-1)^{(n-1)}$ ' grows faster than ' $(n-1)(n-2) \dots$ ' for ' $n \geq 3$ '.
1801
1802    17  18 But ' $n! < n * (n-1)!$ ' is ' $n! < n * (n-1)!$ ', which is true except for ' $n = 1$ '. So is ' $n * (n-1)! < n^{(n-1)}$ '? Not obviously, because for ' $n = 4$ ', ' $4 * 3! = 24$ ' and ' $4^3 = 64$ ', but for ' $n = 3$ ', ' $3! = 6$ ' and ' $3^2 = 9$ '.
1803
1804    19  20 Instead, let's recall the definition of ' $n!$ ' and think about how many times ' $n!$ ' is compared to ' $n^{(n-1)}$ '.
1805
1806    21  22 For ' $n \geq 3$ ', each time you compare ' $n!$ ' with ' $n * (n-1)!$ ', and this is ' $< n * (n-1)! < n * n^{(n-2)} = n^{(n-1)}$ '? Seems wrong because ' $(n-1)! < n^{(n-2)}$ ' would be true for ' $n \geq 2$ ' by induction, which is not a problem since we're proving it.
1807
1808    23  24 Perhaps a better approach is to use telescoping inequalities. Consider ' $n! = n * (n-1) * ... * 2 * 1$ ', and ' $n^{(n-1)}$ ' is the product of ' $n-1$ ' copies of ' $n$ '.
1809
1810    25  26 For ' $n \geq 3$ ', each term in the factorial is smaller than ' $n$ ' (' $k \leq n-1$ ').
1811
1812    27  28 But more precisely, ' $n! = n * (n-1)!$ ' and ' $(n-1)! < n^{(n-2)}$ ' for ' $n \geq 3$ ' (since ' $n-1 \geq 2$ ').
1813
1814    29  30 Let's try to prove ' $(n-1)! < n^{(n-2)}$ '. If ' $(n-1)! < n^{(n-2)}$ ' holds, then ' $n! = n * (n-1)! < n * n^{(n-2)} = n^{(n-1)}$ '.
1815
1816    31  32 But is ' $n * n^{(n-2)} = n^{(n-1)}$ '? Yes, because ' $n * n^{(n-2)} = n^{(1+n-2)} = n^{(n-1)}$ '.
1817
1818    33  34 Thus, ' $n! < n^{(n-1)}$ ', provided that ' $(n-1)! < n^{(n-2)}$ '.
1819
1820    35  36 Wait, ' $n^{(n-2)}$ ' is ' $n * n^{(n-3)}$ ' if ' $n \geq 3$ '.
1821
1822    37

```

```

1836 38 So the induction is:
1837 39 - Base case ' $n = 3$ ': ' $2! = 2$ ' and ' $3^2 = 9$ ', ' $2 < 9$ ' holds.
1838 40 - Inductive step: Assume for ' $n$ ', ' $(n - 1)! < n^{(n - 2)}$ '. Then ' $n$ 
1839 *  $(n - 1)! < n * n^{(n - 2)} = n^{(n - 1)}$ '? But ' $n * n^{(n - 2)} =$ 
1840  $n^{(n - 2 + 1)} = n^{(n - 1)}$ '. So ' $n! = n * (n - 1)! < n^{(n - 1)}$ '.
1841 41 This is a valid induction, provided that ' $(n - 1)! < n^{(n - 2)}$ ' is true for ' $n \geq 3$ '.
1842 42 Now, recall that for ' $n \geq 3$ ':
1843 43 ' $n! = n * (n - 1)!$ ' and ' $(n - 1)! < n^{(n - 2)}$ ' if ' $n \geq 3$ '.
1844 44 But is ' $(n - 1)! < n^{(n - 2)}$ ' true for ' $n \geq 3$ '?
1845 45 For ' $n = 3$ ', LHS = 2, RHS = 1. False. Hmm, ' $3! = 6$ ' and ' $3^{(3 - 3)} = 1$ ? No, ' $3 \geq 3$ ' and ' $3^{(3 - 2)} = 3^{(1)} = 3$ ', ' $2 < 3$ ', so true.
1846 46 Wait no. For ' $n = 4$ ', ' $(3)! = 6$ ', and ' $4^{(4 - 2)} = 4^{(2)} = 16$ ', ' $6 < 16$ ' true.
1847 47 For ' $n = 5$ ', ' $(4)! = 24$ ', ' $5^{(5 - 2)} = 5^3 = 125$ ', ' $24 < 125$ ' true.
1848 48 Hypothesis that ' $(n - 1)! < n^{(n - 2)}$ ' for all ' $n \geq 3$ '.
1849 49 Base case ' $n = 3$ ', ' $2! = 2 < 3^{(1)} = 3$ ' true.
1850 50 Inductive step: Assume ' $(n - 1)! < n^{(n - 2)}$ ' for ' $n \geq 4$ '.
1851 51 We need ' $n! < (n + 1)^{(n - 1)}$ ':
1852 52 ' $n! = n * (n - 1)! < n * n^{(n - 2)} = n^{(n - 1)}$ ', since ' $(n - 1)! < n^{(n - 2)}$ ' holds by the inductive hypothesis for ' $n \geq 3$ '.
1853 53 This works!
1854 54
1855 55 Conclusion: The inequality ' $n! < n^{(n - 1)}$ ' holds for all ' $n \geq 3$ ' (' $n \geq 3$ ').
1856 56
1857 57 But wait, the original Lean statement is ' $n : \mathbb{N}$ ' and ' $3 \leq n$ ', so indices are clear.
1858 58
1859 59 We can proceed directly using induction. The 'have' statements will reflect the proof.
1860 60
1861 61 Lean 4 Proof Sketch:
1862 62
1863 63 1. Base Case (' $n = 3$ '): Check that ' $3! < 3^{(3 - 1)}$ ' or ' $6 < 9$ '.
1864 64 2. Inductive Step: Assume ' $(k)! < k^{(k - 1)}$ ' for some ' $k \geq 3$ ', and prove ' $(k + 1)! < (k + 1)^{(k)}$ ' using the assumption ' $k! < k^{(k - 1)}$ '.
1865 65
1866 66 Abstract Step-by-Step Plan
1867 67
1868 68 1. Base Case (' $n = 3$ '):
1869 69 - Compute ' $3! = 6$ '.
1870 70 - Compute ' $3^{(3 - 1)} = 3^2 = 9$ '.
1871 71 - Since ' $6 < 9$ ', the statement holds.
1872 72
1873 73 2. Inductive Step (' $n \rightarrow n + 1$ '):
1874 74 - Assume ' $n! < n^{(n - 1)}$ ' for some ' $n \geq 3$ ' (induction hypothesis).
1875 75 - Prove ' $(n + 1)! < (n + 1)^n$ ':
1876 76 - ' $(n + 1)! = (n + 1) * n!$ ' (by definition).
1877 77 - By hypothesis, ' $n! < n^{(n - 1)}$ '.
1878 78
1879 79
1880 80
1881 81
1882 82
1883 83
1884 84
1885 85
1886 86
1887 87
1888 88
1889 89

```

```

1890      - Thus, ' $(n + 1)! < (n + 1) * n^{n-1}$ '.
1891      - It remains to prove ' $(n + 1) * n^{n-1} \leq (n + 1)^n$ ',  

1892      i.e., ' $n^{n-1} \leq (n + 1)^{n-1}$ ' (if ' $n + 1 \geq 2$ '? Always true  

1893      since ' $n \geq 2$ ').  

1894      * Since ' $n \geq 2$ ', ' $n^{n-1} \leq (n + 1)^{n-1}$ ' because ' $n \leq$   

1895       $n + 1$ ' and the exponent is ' $n - 1 \geq 2$ ', making the term ' $(n +$   

1896       $1)^{n-1}$ ' strictly greater than ' $n^{n-1}$ '.
1897
1898 93  ##### Lean 4 'have' Statements
1899
1900 95  ````lean4
1901 96  theorem induction
1902 97  (n :  $\mathbb{N}$ )
1903 98  (h0 :  $3 \leq n$ ) :
1904 99  (n)! <  $n^{n-1}$  := by
1905 100 have h_main : (n)! <  $n^{n-1}$  := by
1906 101 sorry
1907 102 sorry
1908 103 ``
1909
1910 105 ##### Explanation of 'have' Statements:
1911 106 1. The 'have' statement 'h_main' directly encodes our goal ' $(n)! < n^{n-1}$ '. Inside the proof, we would perform induction, but in a  

1912      structured Lean proof, we break it into a single 'have h_main'.
1913 107      - The actual proof would expand on 'h_main' by leveraging  

1914      induction, but for this sketch, it suffices to rely on the good  

1915      old induction hypothesis.
1916
1917 108 ##### Complete Lean 4 Proof
1918
1919 109 ````lean4
1920 110 theorem induction
1921 111 (n :  $\mathbb{N}$ )
1922 112 (h0 :  $3 \leq n$ ) :
1923 113 (n)! <  $n^{n-1}$  := by
1924 114 have h_main : (n)! <  $n^{n-1}$  := by
1925 115 have h1 :  $\forall n : \mathbb{N}, 3 \leq n \rightarrow (n)! < n^{n-1}$  := by
1926 116 intro n hn
1927 117 induction' hn with n hn IH
1928 118 · -- Base case:  $n = 3$ 
1929 119 norm_num [Nat.factorial]
1930 120 · -- Inductive step: assume the statement holds for  $n$ , prove
1931 121 for  $n + 1$ 
1932 122 cases n with
1933 123 | zero => contradiction --  $n$  cannot be zero since  $3 \leq n$ 
1934 124 | succ n =>
1935 125 cases n with
1936 126 | zero => contradiction --  $n$  cannot be one since  $3 \leq n$ 
1937 127 | succ n =>
1938 128 cases n with
1939 129 | zero => contradiction --  $n$  cannot be two since  $3 \leq n$ 
1940 130 | succ n =>
1941 131 simp_all [Nat.factorial, Nat.pow_succ, Nat.mul_comm,
1942 132 Nat.mul_assoc, Nat.mul_left_comm]
1943 133 <;>
1944 134 (try omega) <;>
1945 135 (try
1946 136 {
1947 137     nlinarith [pow_pos (by decide : (0 :  $\mathbb{N}$ ) < 2)
1948 138 n, pow_pos (by decide : (0 :  $\mathbb{N}$ ) < 2) (n - 1)]
1949 139     }) <;>
1950 140 (try
1951 141 {
1952 142     ring_nf at *
1953 143     <;>

```

```

1944                               nlinarith [pow_pos (by decide : (0 : N) < 2)
1945   n, pow_pos (by decide : (0 : N) < 2) (n - 1)]
1946   ) <;>
1947   (try
1948   {
1949     simp_all [Nat.factorial_succ, Nat.mul_comm,
1950     Nat.mul_assoc, Nat.mul_left_comm]
1951     <;>
1952     norm_num at *
1953   <;>
1954   ring_nf at *
1955   <;>
1956   nlinarith [pow_pos (by decide : (0 : N) < 2)
1957   n, pow_pos (by decide : (0 : N) < 2) (n - 1)]
1958   ) )
1959   exact h1 n h0
1960   exact h_main
1961   ```

```

E.2 CASE STUDY OF SUCCESSFUL EXAMPLE WITH ITERATIVE REFINEMENT

E.2.1 DETAILED ANALYSIS

We analyze in detail the reasoning process and the output for the problem `mathd_algebra_275`, a case where direct proof without iterative refinement failed, but iterative refinement succeeded after three iterations. This problem asks to find the value of the expression $(11^{1/4})^{6x+2}$ given the equation $(11^{1/4})^{3x-3} = 1/5$.

We analyze the final successful iteration of the iterative refinement process for this problem. The prompt used in this final iteration along with the corresponding output is shown in Appendix E.2.4.

In this case, the input prompt highlights two failures: a `linarith` error and an `unsolved goals` state. Both errors originated from the model's initial attempt to resolve complex non-linear expressions using standard automated tactics, which were insufficient for the structural complexity involved. Crucially, the model interpreted these error messages as indicators of the limitations of the automated tools. Consequently, instead of attempting superficial fixes, the model adopted a fundamentally more robust mathematical strategy. This demonstrates how explicit feedback regarding the boundaries of automated proving effectively guides the model toward a successful resolution. Below, we analyze the failures in detail, explaining their root causes and how the final successful proof overcomes them.

The first Lean error message is as follows (as shown in the prompt used in the final refinement step):

linarith failed to find a contradiction

The goal state at the point of failure involved complex nested exponentiation of real numbers, specifically terms such as $((11^{1/4})^{3x-3})^2$. The failure stems from the misapplication of a linear arithmetic solver to a fundamentally non-linear problem. In this instance, the validity of the equality relied on the algebraic properties of exponentiation, specifically the power rule $(a^b)^c = a^{bc}$. However, linarith does not have built-in knowledge of these non-linear identities. Because the solver could not peer inside the `Real.rpow` terms to see that the left-hand side and right-hand side were algebraically equivalent, it treated them as distinct, unrelated variables, thus failing to derive the necessary contradiction.

Upon receiving this error message, the model declares its intention to fix the code on line 3, and immediately proceeds to analyze this first error in the “Observations” section on line 10. Here, the model devises a corrective strategy that switches to applying the natural logarithm (`Real.log`) to both sides, instead of attempting to manipulate the exponents directly (which leads to the non-linear structures that baffled `linarith`). This transformation converts the exponentiation operations into multiplication, and the problem is mapped from a non-linear domain into a linear domain where the constraints on x become simple linear equations. The model elaborates on the details of this logarithmic strategy in the “Rewriting the Hypothesis” section, starting from line 11.

1998 The second error, flagged as unsolved goals, arose from the tactics' inability to verify a commutativity relation ($A \times B = B \times A$) involving complex nested exponential terms. The standard ring tactic
 1999 failed to normalize these opaque structures, leaving the proof in an incomplete state. The resolution
 2000 of this error emerges organically as a byproduct of the primary strategy shift. Consequently, while
 2001 the "Observations" section commencing on line 10 does not explicitly address this specific error, the
 2002 model naturally analyzes the details and resolves the issue during the formulation of the corrected
 2003 version, specifically within the "Rewriting the Goal" section (starting on line 16) and the "Correct
 2004 Approach" section (starting on line 46).
 2005

2006 For comparison, Appendix E.2.5 presents the output generated without iterative refinement. As ob-
 2007 served in this output, due to the absence of error feedback, the model makes no attempt to resolve
 2008 specific failure points; instead, its activity is limited to formulating a general strategy and construct-
 2009 ing the Lean proof.

2010

2011 E.2.2 LEAN ENVIRONMENT SETUP

2012

2013 All Lean code is executed in exactly the same environment as described in Appendix E.1.2. Each
 2014 code snippet is preceded by the same Lean header shown in Appendix E.1.2, which we omit in the
 2015 subsequent examples for conciseness.

2016

2017 E.2.3 THE PROBLEM

2018

2019 This problem asks you to find the value of the expression $(11^{1/4})^{6x+2}$ given the equation
 2020 $(11^{1/4})^{3x-3} = 1/5$:

2021

```
1 theorem mathd_algebra_275 (x : ℝ) (h : ((11 : ℝ) ^ (1 / 4 : ℝ)) ^  

2   (3 * x - 3) = 1 / 5) :  

3   ((11 : ℝ) ^ (1 / 4 : ℝ)) ^ (6 * x + 2) = 121 / 25 := by sorry
```

2024

2025

2026 E.2.4 FINAL ITERATIVE REFINEMENT STEP

2027

2028 This problem could not be solved by direct proving, but it succeeds after three iterations of iter-
 2029 ative refinement. Below, we show the prompt used in the final refinement step and the model's
 2030 corresponding output.

2031

2031 The prompt used in the final refinement step is as follows:

2032

2033

```
1 Your goal is to implement the following theorem, using Lean 4 and  

2   the mathlib library:  

3  

4   ```:lean4  

5   import Mathlib  

6   import Aesop  

7  

8   set_option maxHeartbeats 0  

9  

10  open BigOperators Real Nat Topology Rat  

11  

12  theorem mathd_algebra_275 (x : ℝ) (h : ((11 : ℝ) ^ (1 / 4 : ℝ)) ^  

13    (3 * x - 3) = 1 / 5) :  

14    ((11 : ℝ) ^ (1 / 4 : ℝ)) ^ (6 * x + 2) = 121 / 25 := by  

15    ````  

16  

17  Your proof is as follows:  

18  ```:lean4  

19  import Mathlib  

20  import Aesop  

21  set_option maxHeartbeats 0  

22
```

```

2052 23 open BigOperators Real Nat Topology Rat
2053 24
2054 25
2055 26 theorem mathd_algebra_275 (x : ℝ) (h : ((11 : ℝ) ^ (1 / 4 : ℝ)) ^ (3 * x - 3) = 1 / 5) :
2056 27      ((11 : ℝ) ^ (1 / 4 : ℝ)) ^ (6 * x + 2) = 121 / 25 := by
2057 28      have h1 : ((11 : ℝ) ^ (1 / 4 : ℝ)) ^ (6 * x + 2) = ((11 : ℝ) ^ (1 / 4 : ℝ)) ^ (2 * (3 * x - 3) + 8) := by
2058 29          have h1 : (6 * x + 2 : ℝ) = 2 * (3 * x - 3) + 8 := by
2059 30              ring_nf
2060 31          <;>
2061 32          linarith
2062 33      rw [h1]
2063 34      <;>
2064 35      norm_num
2065 36      <;>
2066 37      linarith
2067 38
2068 39      have h2 : ((11 : ℝ) ^ (1 / 4 : ℝ)) ^ (2 * (3 * x - 3) + 8) =
2069 40          (((11 : ℝ) ^ (1 / 4 : ℝ)) ^ (3 * x - 3)) ^ 2 * ((11 : ℝ) ^ (1 / 4 : ℝ)) ^ 8 := by
2070 41          have h3 : ((11 : ℝ) ^ (1 / 4 : ℝ)) ^ (2 * (3 * x - 3) + 8) =
2071 42          (((11 : ℝ) ^ (1 / 4 : ℝ)) ^ (2 * (3 * x - 3)) * ((11 : ℝ) ^ (1 / 4 : ℝ)) ^ 8) := by
2072 43          have h4 : ((11 : ℝ) ^ (1 / 4 : ℝ)) ^ (2 * (3 * x - 3) + 8) =
2073 44          (((11 : ℝ) ^ (1 / 4 : ℝ)) ^ (2 * (3 * x - 3)) * ((11 : ℝ) ^ (1 / 4 : ℝ)) ^ 8) := by
2074 45              rw [show (2 * (3 * x - 3) + 8 : ℝ) = (2 * (3 * x - 3) : ℝ) + 8 by ring]
2075 46                  rw [Real.rpow_add (by positivity)]
2076 47          <;> ring_nf
2077 48          <;> norm_num
2078 49          <;> linarith
2079 50          exact h4
2080 51      rw [h3]
2081 52          have h5 : ((11 : ℝ) ^ (1 / 4 : ℝ)) ^ (2 * (3 * x - 3)) = (((11 : ℝ) ^ (1 / 4 : ℝ)) ^ (3 * x - 3)) ^ 2 := by
2082 53          have h6 : ((11 : ℝ) ^ (1 / 4 : ℝ)) ^ (2 * (3 * x - 3)) = (((11 : ℝ) ^ (1 / 4 : ℝ)) ^ (3 * x - 3)) ^ 2 := by
2083 54          have h7 : (2 * (3 * x - 3) : ℝ) = (3 * x - 3) + (3 * x - 3) := by ring
2084 55          rw [h7]
2085 56          have h8 : ((11 : ℝ) ^ (1 / 4 : ℝ)) ^ ((3 * x - 3) + (3 * x - 3)) = ((11 : ℝ) ^ (1 / 4 : ℝ)) ^ (3 * x - 3) := by
2086 57              rw [Real.rpow_add (by positivity)]
2087 58          <;> ring_nf
2088 59          <;> norm_num
2089 60          <;> linarith
2090 61      rw [h8]
2091 62          <;> ring_nf
2092 63          <;> field_simp
2093 64          <;> ring_nf
2094 65          <;> norm_num
2095 66          <;> linarith
2096 67
2097 68
2098 69
2099 70      have h3 : (((11 : ℝ) ^ (1 / 4 : ℝ)) ^ (3 * x - 3)) ^ 2 = (1 / 5 : ℝ) ^ 2 := by
2100 71          have h3 : ((11 : ℝ) ^ (1 / 4 : ℝ)) ^ (3 * x - 3) = 1 / 5 := h
2101 72          rw [h3]

```

```

2106      73      <;> norm_num
2107      74
2108      75      have h4 : ((11 : ℝ) ^ (1 / 4 : ℝ)) ^ 8 = 121 := by
2109      76          have h4 : ((11 : ℝ) ^ (1 / 4 : ℝ)) ^ 8 = 121 := by
2110      77              have h5 : ((11 : ℝ) ^ (1 / 4 : ℝ)) ^ 8 = (11 : ℝ) ^ ((1 / 4
2111      78                  : ℝ) * 8) := by
2112          79              rw [← Real.rpow_nat_cast]
2113          80              rw [← Real.rpow_mul] <;> ring_nf <;> norm_num
2114          81                  <;> positivity
2115          82                      rw [h5]
2116          83              have h6 : (11 : ℝ) ^ ((1 / 4 : ℝ) * 8) = (11 : ℝ) ^ (2 : ℝ)
2117          84                  := by
2118          85                      norm_num
2119          86                      <;> ring_nf
2120          87                      <;> field_simp
2121          88                      <;> ring_nf
2122          89                      rw [h6]
2123          90              have h7 : (11 : ℝ) ^ (2 : ℝ) = 121 := by
2124          91                  norm_num [Real.rpow_two]
2125          92                  <;> ring_nf
2126          93                  <;> norm_num
2127          94                      exact h4
2128
2129          95      have h5 : ((11 : ℝ) ^ (1 / 4 : ℝ)) ^ (6 * x + 2) = 121 / 25 := by
2130          96          calc
2131          97              ((11 : ℝ) ^ (1 / 4 : ℝ)) ^ (6 * x + 2) = ((11 : ℝ) ^ (1 / 4
2132          98                  : ℝ)) ^ (2 * (3 * x - 3) + 8) := by rw [h1]
2133          99                  _ = (((11 : ℝ) ^ (1 / 4 : ℝ)) ^ (3 * x - 3)) ^ 2 * ((11 : ℝ)
2134          100                  ^ (1 / 4 : ℝ)) ^ 8 := by rw [h2]
2135          101                  _ = (1 / 5 : ℝ) ^ 2 * ((11 : ℝ) ^ (1 / 4 : ℝ)) ^ 8 := by rw
2136          102                  [h3]
2137          103                  _ = (1 / 5 : ℝ) ^ 2 * 121 := by rw [h4]
2138          104                  _ = 121 / 25 := by
2139          105                      norm_num
2140          106                      <;> ring_nf
2141          107                      <;> norm_num
2142          108                      <;> linarith
2143          109
2144          110      ```

2145      111      The proof failed to compile with errors.
2146      112      The error occurred at the following line(s):
2147      113
2148      114      Error line (line 29):
2149      115      ````lean4
2150      116          <;> linarith
2151      117          ``
2152      118
2153      119      Error message:
2154      120      ````lean4
2155      121      linarith failed to find a contradiction
2156      122      case h1.h
2157      123      x : ℝ
2158      124      h : (11 ^ (1 / 4)) ^ (3 * x - 3) = 1 / 5
2159      125      h1 : (11 ^ (1 / 4)) ^ (6 * x + 2) = (11 ^ (1 / 4)) ^ (2 * (3 * x -
3) + 8)
2160      126      a : (11 ^ (1 / 4)) ^ (-6 + x * 6) * (11 ^ (1 / 4)) ^ 8 < (11 ^ (1
/ 4)) ^ 8 * (11 ^ (1 / 4)) ^ (-6 + x * 6)
2161      127      ⊢ False
2162      128      failed
2163      129      ``
2164
2165
2166
2167
2168
2169
2170
2171
2172
2173
2174
2175
2176
2177
2178
2179
2180
2181
2182
2183
2184
2185
2186
2187
2188
2189
2190
2191
2192
2193
2194
2195
2196
2197
2198
2199
2200
2201
2202
2203
2204
2205
2206
2207
2208
2209
2210
2211
2212
2213
2214
2215
2216
2217
2218
2219
2220
2221
2222
2223
2224
2225
2226
2227
2228
2229
2230
2231
2232
2233
2234
2235
2236
2237
2238
2239
2240
2241
2242
2243
2244
2245
2246
2247
2248
2249
2250
2251
2252
2253
2254
2255
2256
2257
2258
2259
2260
2261
2262
2263
2264
2265
2266
2267
2268
2269
2270
2271
2272
2273
2274
2275
2276
2277
2278
2279
2280
2281
2282
2283
2284
2285
2286
2287
2288
2289
2290
2291
2292
2293
2294
2295
2296
2297
2298
2299
2300
2301
2302
2303
2304
2305
2306
2307
2308
2309
2310
2311
2312
2313
2314
2315
2316
2317
2318
2319
2320
2321
2322
2323
2324
2325
2326
2327
2328
2329
2330
2331
2332
2333
2334
2335
2336
2337
2338
2339
2340
2341
2342
2343
2344
2345
2346
2347
2348
2349
2350
2351
2352
2353
2354
2355
2356
2357
2358
2359
2360
2361
2362
2363
2364
2365
2366
2367
2368
2369
2370
2371
2372
2373
2374
2375
2376
2377
2378
2379
2380
2381
2382
2383
2384
2385
2386
2387
2388
2389
2390
2391
2392
2393
2394
2395
2396
2397
2398
2399
2400
2401
2402
2403
2404
2405
2406
2407
2408
2409
2410
2411
2412
2413
2414
2415
2416
2417
2418
2419
2420
2421
2422
2423
2424
2425
2426
2427
2428
2429
2430
2431
2432
2433
2434
2435
2436
2437
2438
2439
2440
2441
2442
2443
2444
2445
2446
2447
2448
2449
2450
2451
2452
2453
2454
2455
2456
2457
2458
2459
2460
2461
2462
2463
2464
2465
2466
2467
2468
2469
2470
2471
2472
2473
2474
2475
2476
2477
2478
2479
2480
2481
2482
2483
2484
2485
2486
2487
2488
2489
2490
2491
2492
2493
2494
2495
2496
2497
2498
2499
2500
2501
2502
2503
2504
2505
2506
2507
2508
2509
2510
2511
2512
2513
2514
2515
2516
2517
2518
2519
2520
2521
2522
2523
2524
2525
2526
2527
2528
2529
2530
2531
2532
2533
2534
2535
2536
2537
2538
2539
2540
2541
2542
2543
2544
2545
2546
2547
2548
2549
2550
2551
2552
2553
2554
2555
2556
2557
2558
2559
2560
2561
2562
2563
2564
2565
2566
2567
2568
2569
2570
2571
2572
2573
2574
2575
2576
2577
2578
2579
2580
2581
2582
2583
2584
2585
2586
2587
2588
2589
2590
2591
2592
2593
2594
2595
2596
2597
2598
2599
2600
2601
2602
2603
2604
2605
2606
2607
2608
2609
2610
2611
2612
2613
2614
2615
2616
2617
2618
2619
2620
2621
2622
2623
2624
2625
2626
2627
2628
2629
2630
2631
2632
2633
2634
2635
2636
2637
2638
2639
2640
2641
2642
2643
2644
2645
2646
2647
2648
2649
2650
2651
2652
2653
2654
2655
2656
2657
2658
2659
2660
2661
2662
2663
2664
2665
2666
2667
2668
2669
2670
2671
2672
2673
2674
2675
2676
2677
2678
2679
2680
2681
2682
2683
2684
2685
2686
2687
2688
2689
2690
2691
2692
2693
2694
2695
2696
2697
2698
2699
2700
2701
2702
2703
2704
2705
2706
2707
2708
2709
2710
2711
2712
2713
2714
2715
2716
2717
2718
2719
2720
2721
2722
2723
2724
2725
2726
2727
2728
2729
2730
2731
2732
2733
2734
2735
2736
2737
2738
2739
2740
2741
2742
2743
2744
2745
2746
2747
2748
2749
2750
2751
2752
2753
2754
2755
2756
2757
2758
2759
2760
2761
2762
2763
2764
2765
2766
2767
2768
2769
2770
2771
2772
2773
2774
2775
2776
2777
2778
2779
2780
2781
2782
2783
2784
2785
2786
2787
2788
2789
2790
2791
2792
2793
2794
2795
2796
2797
2798
2799
2800
2801
2802
2803
2804
2805
2806
2807
2808
2809
2810
2811
2812
2813
2814
2815
2816
2817
2818
2819
2820
2821
2822
2823
2824
2825
2826
2827
2828
2829
2830
2831
2832
2833
2834
2835
2836
2837
2838
2839
2840
2841
2842
2843
2844
2845
2846
2847
2848
2849
2850
2851
2852
2853
2854
2855
2856
2857
2858
2859
2860
2861
2862
2863
2864
2865
2866
2867
2868
2869
2870
2871
2872
2873
2874
2875
2876
2877
2878
2879
2880
2881
2882
2883
2884
2885
2886
2887
2888
2889
2890
2891
2892
2893
2894
2895
2896
2897
2898
2899
2900
2901
2902
2903
2904
2905
2906
2907
2908
2909
2910
2911
2912
2913
2914
2915
2916
2917
2918
2919
2920
2921
2922
2923
2924
2925
2926
2927
2928
2929
2930
2931
2932
2933
2934
2935
2936
2937
2938
2939
2940
2941
2942
2943
2944
2945
2946
2947
2948
2949
2950
2951
2952
2953
2954
2955
2956
2957
2958
2959
2960
2961
2962
2963
2964
2965
2966
2967
2968
2969
2970
2971
2972
2973
2974
2975
2976
2977
2978
2979
2980
2981
2982
2983
2984
2985
2986
2987
2988
2989
2990
2991
2992
2993
2994
2995
2996
2997
2998
2999
2999

```

```

2160
2161
2162 Error line (line 24):
2163   ````lean4
2164     have h4 : ((11 : ℝ) ^ (1 / 4 : ℝ)) ^ (2 * (3 * x - 3) + 8) =
2165     ((11 : ℝ) ^ (1 / 4 : ℝ)) ^ (2 * (3 * x - 3)) * ((11 : ℝ) ^ (1 / 4
2166     : ℝ)) ^ 8 := by
2167       rw [show (2 * (3 * x - 3) + 8 : ℝ) = (2 * (3 * x - 3) : ℝ)
2168       + 8 by ring]
2169       rw [Real.rpow_add (by positivity)]
2170       < ; > ring_nf
2171       < ; > norm_num
2172       < ; > linarith
2173   `````
2174 Error message:
2175   ````lean4
2176 unsolved goals
2177 x : ℝ
2178 h : (11 ^ (1 / 4)) ^ (3 * x - 3) = 1 / 5
2179 h1 : (11 ^ (1 / 4)) ^ (6 * x + 2) = (11 ^ (1 / 4)) ^ (2 * (3 * x -
2180 3) + 8)
2181   ⊢ (11 ^ (1 / 4)) ^ (-6 + x * 6) * (11 ^ (1 / 4)) ^ 8 = (11 ^ (1 /
2182 4)) ^ 8 * (11 ^ (1 / 4)) ^ (-6 + x * 6)
2183   `````
2184
2185
2186
2187 Fix these errors and complete the following Lean 4 code:
2188
2189   ````lean4
2190   import Mathlib
2191   import Aesop
2192
2193   set_option maxHeartbeats 0
2194
2195   open BigOperators Real Nat Topology Rat
2196
2197   theorem mathd_algebra_275 (x : ℝ) (h : ((11 : ℝ) ^ (1 / 4 : ℝ)) ^ (3 * x - 3) = 1 / 5) :
2198     ((11 : ℝ) ^ (1 / 4 : ℝ)) ^ (6 * x + 2) = 121 / 25 := by
2199   `````
2200
2201 Before producing the Lean 4 code to formally prove the given
2202 theorem, provide a detailed proof plan outlining the main proof
2203 steps and strategies.
2204 The plan should highlight key ideas, intermediate lemmas, and
2205 proof structures that will guide the construction of the final
2206 formal proof.

```

The corresponding output from the prover model is as follows:

```
2202
2203 1 ## Detailed Proof and Analysis
2204 2
2205 3 First, let's carefully analyze the given problem and the Lean 4
2206 statement to understand where the errors might have occurred.
2207 4
2208 5 ##### Problem Breakdown:
2209 6 We are given:
2210 7 1.  $((11 : \mathbb{R}) ^ (1 / 4 : \mathbb{R})) ^ (3 * x - 3) = 1 / 5$ 
2211 8 2. We need to prove that  $((11 : \mathbb{R}) ^ (1 / 4 : \mathbb{R})) ^ (6 * x + 2) =$ 
2212 9  $121 / 25$ .
2213 10 ##### Observations:
2214 11 1. The expression  $((11 : \mathbb{R}) ^ (1 / 4 : \mathbb{R}))$  is a real number raised
2215 to the power of  $1/4$ , which is not a standard exponentiation
```

2214 operation in Lean 4. In Lean 4, ' $a ^ b$ ' for ' $a : \mathbb{R}$ ' and ' $b : \mathbb{R}$ ' is
 2215 interpreted as 'Real.rpow a b', i.e., ' $a ^ b = \exp(b * \log a)$ '.
 2216 2. The exponent ' $3 * x - 3$ ' is real-valued, and the exponentiation
 2217 is 'Real.rpow' (real power). Similarly, ' $6 * x + 2$ ' is real-valued.
 2218 3. The hypothesis ' h ' is ' $((11 : \mathbb{R}) ^ (1 / 4 : \mathbb{R})) ^ (3 * x - 3) =$
 2219 $1 / 5$ ', which is not a standard exponentiation operation unless we
 2220 are interpreting '^' as 'Real.rpow'.
 2221 4. The Lean 4 code uses 'Real.rpow' implicitly, and ' h ' is a
 2222 statement about real powers.
 2223
 2224 **#### Rewriting the Hypothesis:**
 2225 Given ' $h : ((11 : \mathbb{R}) ^ (1 / 4 : \mathbb{R})) ^ (3 * x - 3) = 1 / 5$ ', we can
 2226 take the natural logarithm of both sides to find a relationship
 2227 involving ' x '.
 2228
 2229 First, note that:
 2230 ' $(11 : \mathbb{R}) ^ (1 / 4 : \mathbb{R}) > 0$ ', so the logarithm is well-defined.
 2231
 2232 Taking the natural logarithm of both sides of ' h ':
 2233 ' $\log((11 : \mathbb{R}) ^ (1 / 4 : \mathbb{R})) ^ (3 * x - 3) = \log(1 / 5)$ '.
 2234
 2235 Using the logarithm power rule:
 2236 ' $(3 * x - 3) * \log((11 : \mathbb{R}) ^ (1 / 4 : \mathbb{R})) = \log(1 / 5)$ '.
 2237
 2238 Since ' $\log(1 / 5) = -\log 5$ ', we have:
 2239 ' $(3 * x - 3) * \log((11 : \mathbb{R}) ^ (1 / 4 : \mathbb{R})) = -\log 5$ '.
 2240
 2241 But ' $\log((11 : \mathbb{R}) ^ (1 / 4 : \mathbb{R})) = (1 / 4) * \log 11$ ', so:
 2242 ' $(3 * x - 3) * (1 / 4) * \log 11 = -\log 5$ '.
 2243
 2244 Multiply both sides by 4:
 2245 ' $(3 * x - 3) * \log 11 = -4 * \log 5$ '.
 2246
 2247 Divide both sides by ' $\log 11$ ' (assuming ' $\log 11 \neq 0$ ', which is
 2248 true since ' $11 > 1$ '):
 2249 ' $3 * x - 3 = -4 * (\log 5 / \log 11)$ '.
 2250
 2251 Thus:
 2252 ' $3 * x = 3 - 4 * (\log 5 / \log 11)$ '.
 2253
 2254 Therefore:
 2255 ' $x = 1 - (4 / 3) * (\log 5 / \log 11)$ '.
 2256
 2257 **#### Rewriting the Goal:**
 2258 We need to find ' $((11 : \mathbb{R}) ^ (1 / 4 : \mathbb{R})) ^ (6 * x + 2)$ '.
 2259
 2260 First, note that:
 2261 ' $6 * x + 2 = 6 * (1 - (4 / 3) * (\log 5 / \log 11)) + 2 = 6 - 8 *$
 2262 $(\log 5 / \log 11) + 2 = 8 - 8 * (\log 5 / \log 11)$ '.
 2263
 2264 Thus:
 2265 ' $6 * x + 2 = 8 * (1 - (\log 5 / \log 11))$ '.
 2266
 2267 Therefore:
 2268 ' $((11 : \mathbb{R}) ^ (1 / 4 : \mathbb{R})) ^ (6 * x + 2) = ((11 : \mathbb{R}) ^ (1 / 4 : \mathbb{R})$
 2269 $) ^ (8 * (1 - (\log 5 / \log 11)))$ '.
 2270
 2271 But ' $((11 : \mathbb{R}) ^ (1 / 4 : \mathbb{R})) ^ (8 * (1 - (\log 5 / \log 11))) =$
 2272 ' $((11 : \mathbb{R}) ^ (1 / 4 : \mathbb{R})) ^ 8 * ((11 : \mathbb{R}) ^ (1 / 4 : \mathbb{R})) ^ (-8 *$
 2273 $(\log 5 / \log 11))$ '.
 2274
 2275 But ' $((11 : \mathbb{R}) ^ (1 / 4 : \mathbb{R})) ^ 8 = (11 : \mathbb{R}) ^ 2 = 121$ ', and ' $((11$
 2276 $: \mathbb{R}) ^ (1 / 4 : \mathbb{R})) ^ (-8 * (\log 5 / \log 11)) = 1 / ((11 : \mathbb{R}) ^ (1$
 2277 $/ 4 : \mathbb{R})) ^ (8 * (\log 5 / \log 11))$ '.
 2278
 2279 61

```

2268 62 But  $((11 : \mathbb{R})^{(1 / 4 : \mathbb{R})})^{(8 * (\log 5 / \log 11))} = (11 : \mathbb{R})$ 
2269  $\wedge (8 * (\log 5 / \log 11) * (1 / 4)) = (11 : \mathbb{R})^{(2 * (\log 5 / \log 11))}$ 
2270  $= (11 : \mathbb{R})^{(\log 5 / \log 11 * 2)} = \exp(\log 5 / \log 11 * 2 * \log 11) = \exp(2 * \log 5) = \exp(\log 25) = 25$ .
2271
2272 63
2273 64 Thus:
2274 65  $((11 : \mathbb{R})^{(1 / 4 : \mathbb{R})})^{(6 * x + 2)} = 121 * (1 / 25) = 121 / 25$ .
2275
2276 66
2277 67 However, this is not correct because  $((11 : \mathbb{R})^{(1 / 4 : \mathbb{R})})^{(8 * (\log 5 / \log 11))}$  is not '25'.
2278
2279 68 #### Correct Approach:
2280 69 We can directly compute  $((11 : \mathbb{R})^{(1 / 4 : \mathbb{R})})^{(6 * x + 2)}$  using the relationship we derived for 'x'.
2281
2282 70 Recall that  $x = 1 - (4 / 3) * (\log 5 / \log 11)$ .
2283
2284 71 Thus:
2285 72  $6 * x + 2 = 6 * (1 - (4 / 3) * (\log 5 / \log 11)) + 2 = 6 - 8 * (\log 5 / \log 11) + 2 = 8 - 8 * (\log 5 / \log 11)$ .
2286
2287 73 But  $8 - 8 * (\log 5 / \log 11) = 8 * (1 - (\log 5 / \log 11)) = 8 * (\log(11 / 5) / \log 11)$ .
2288
2289 74 But  $((11 : \mathbb{R})^{(1 / 4 : \mathbb{R})})^{(6 * x + 2)} = ((11 : \mathbb{R})^{(1 / 4 : \mathbb{R})})^{(8 * (\log(11 / 5) / \log 11))} = ((11 : \mathbb{R})^{(1 / 4 : \mathbb{R})})^{(8 * (\log(11 / 5) / \log 11))} = (11 : \mathbb{R})^{(2 * (\log(11 / 5) / \log 11))} = (11 : \mathbb{R})^{(\log(11 / 5) / (\log 11 / 2))} = \exp(\log(11 / 5) / (\log 11 / 2) * \log 11) = \exp(2 * \log(11 / 5)) = \exp(\log((11 / 5) ^ 2)) = (11 / 5) ^ 2 = 121 / 25$ .
2290
2291
2292
2293
2294
2295
2296 75 But wait,  $((11 : \mathbb{R})^{(1 / 4 : \mathbb{R})})^{(6 * x + 2)} = ((11 : \mathbb{R})^{(1 / 4 : \mathbb{R})})^{(8 * (\log(11 / 5) / \log 11))} = (11 : \mathbb{R})^{((1 / 4 : \mathbb{R}) * 8 * (\log(11 / 5) / \log 11))} = (11 : \mathbb{R})^{(2 * (\log(11 / 5) / \log 11))} = (11 : \mathbb{R})^{(\log(11 / 5) / (\log 11 / 2))} = \exp(\log(11 / 5) / (\log 11 / 2) * \log 11) = \exp(2 * \log(11 / 5)) = (11 / 5) ^ 2 = 121 / 25$ .
2297
2298
2299
2300
2301
2302
2303
2304 76 This is correct because  $(11 / 5) ^ 2 = 121 / 25$ .
2305
2306
2307
2308
2309
2310
2311
2312
2313
2314
2315
2316
2317
2318
2319
2320
2321
2322
2323
2324
2325
2326
2327
2328
2329
2330
2331
2332
2333
2334
2335
2336
2337
2338
2339
2340
2341
2342
2343
2344
2345
2346
2347
2348
2349
2350
2351
2352
2353
2354
2355
2356
2357
2358
2359
2360
2361
2362
2363
2364
2365
2366
2367
2368
2369
2370
2371
2372
2373
2374
2375
2376
2377
2378
2379
2380
2381
2382
2383
2384
2385
2386
2387
2388
2389
2390
2391
2392
2393
2394
2395
2396
2397
2398
2399
2400
2401
2402
2403
2404
2405
2406
2407
2408
2409
2410
2411
2412
2413
2414
2415
2416
2417
2418
2419
2420
2421
2422
2423
2424
2425
2426
2427
2428
2429
2430
2431
2432
2433
2434
2435
2436
2437
2438
2439
2440
2441
2442
2443
2444
2445
2446
2447
2448
2449
2450
2451
2452
2453
2454
2455
2456
2457
2458
2459
2460
2461
2462
2463
2464
2465
2466
2467
2468
2469
2470
2471
2472
2473
2474
2475
2476
2477
2478
2479
2480
2481
2482
2483
2484
2485
2486
2487
2488
2489
2490
2491
2492
2493
2494
2495
2496
2497
2498
2499
2500
2501
2502
2503
2504
2505
2506
2507
2508
2509
2510
2511
2512
2513
2514
2515
2516
2517
2518
2519
2520
2521
2522
2523
2524
2525
2526
2527
2528
2529
2530
2531
2532
2533
2534
2535
2536
2537
2538
2539
2540
2541
2542
2543
2544
2545
2546
2547
2548
2549
2550
2551
2552
2553
2554
2555
2556
2557
2558
2559
2560
2561
2562
2563
2564
2565
2566
2567
2568
2569
2570
2571
2572
2573
2574
2575
2576
2577
2578
2579
2580
2581
2582
2583
2584
2585
2586
2587
2588
2589
2590
2591
2592
2593
2594
2595
2596
2597
2598
2599
2600
2601
2602
2603
2604
2605
2606
2607
2608
2609
2610
2611
2612
2613
2614
2615
2616
2617
2618
2619
2620
2621
2622
2623
2624
2625
2626
2627
2628
2629
2630
2631
2632
2633
2634
2635
2636
2637
2638
2639
2640
2641
2642
2643
2644
2645
2646
2647
2648
2649
2650
2651
2652
2653
2654
2655
2656
2657
2658
2659
2660
2661
2662
2663
2664
2665
2666
2667
2668
2669
2670
2671
2672
2673
2674
2675
2676
2677
2678
2679
2680
2681
2682
2683
2684
2685
2686
2687
2688
2689
2690
2691
2692
2693
2694
2695
2696
2697
2698
2699
2700
2701
2702
2703
2704
2705
2706
2707
2708
2709
2710
2711
2712
2713
2714
2715
2716
2717
2718
2719
2720
2721
2722
2723
2724
2725
2726
2727
2728
2729
2730
2731
2732
2733
2734
2735
2736
2737
2738
2739
2740
2741
2742
2743
2744
2745
2746
2747
2748
2749
2750
2751
2752
2753
2754
2755
2756
2757
2758
2759
2760
2761
2762
2763
2764
2765
2766
2767
2768
2769
2770
2771
2772
2773
2774
2775
2776
2777
2778
2779
2780
2781
2782
2783
2784
2785
2786
2787
2788
2789
2790
2791
2792
2793
2794
2795
2796
2797
2798
2799
2800
2801
2802
2803
2804
2805
2806
2807
2808
2809
2810
2811
2812
2813
2814
2815
2816
2817
2818
2819
2820
2821
2822
2823
2824
2825
2826
2827
2828
2829
2830
2831
2832
2833
2834
2835
2836
2837
2838
2839
2840
2841
2842
2843
2844
2845
2846
2847
2848
2849
2850
2851
2852
2853
2854
2855
2856
2857
2858
2859
2860
2861
2862
2863
2864
2865
2866
2867
2868
2869
2870
2871
2872
2873
2874
2875
2876
2877
2878
2879
2880
2881
2882
2883
2884
2885
2886
2887
2888
2889
2890
2891
2892
2893
2894
2895
2896
2897
2898
2899
2900
2901
2902
2903
2904
2905
2906
2907
2908
2909
2910
2911
2912
2913
2914
2915
2916
2917
2918
2919
2920
2921
2922
2923
2924
2925
2926
2927
2928
2929
2930
2931
2932
2933
2934
2935
2936
2937
2938
2939
2940
2941
2942
2943
2944
2945
2946
2947
2948
2949
2950
2951
2952
2953
2954
2955
2956
2957
2958
2959
2960
2961
2962
2963
2964
2965
2966
2967
2968
2969
2970
2971
2972
2973
2974
2975
2976
2977
2978
2979
2980
2981
2982
2983
2984
2985
2986
2987
2988
2989
2990
2991
2992
2993
2994
2995
2996
2997
2998
2999
3000
3001
3002
3003
3004
3005
3006
3007
3008
3009
3010
3011
3012
3013
3014
3015
3016
3017
3018
3019
3020
3021
3022
3023
3024
3025
3026
3027
3028
3029
3030
3031
3032
3033
3034
3035
3036
3037
3038
3039
3040
3041
3042
3043
3044
3045
3046
3047
3048
3049
3050
3051
3052
3053
3054
3055
3056
3057
3058
3059
3060
3061
3062
3063
3064
3065
3066
3067
3068
3069
3070
3071
3072
3073
3074
3075
3076
3077
3078
3079
3080
3081
3082
3083
3084
3085
3086
3087
3088
3089
3090
3091
3092
3093
3094
3095
3096
3097
3098
3099
3100
3101
3102
3103
3104
3105
3106
3107
3108
3109
3110
3111
3112
3113
3114
3115
3116
3117
3118
3119
3120
3121
3122
3123
3124
3125
3126
3127
3128
3129
3130
3131
3132
3133
3134
3135
3136
3137
3138
3139
3140
3141
3142
3143
3144
3145
3146
3147
3148
3149
3150
3151
3152
3153
3154
3155
3156
3157
3158
3159
3160
3161
3162
3163
3164
3165
3166
3167
3168
3169
3170
3171
3172
3173
3174
3175
3176
3177
3178
3179
3180
3181
3182
3183
3184
3185
3186
3187
3188
3189
3190
3191
3192
3193
3194
3195
3196
3197
3198
3199
3200
3201
3202
3203
3204
3205
3206
3207
3208
3209
3210
3211
3212
3213
3214
3215
3216
3217
3218
3219
3220
3221
3222
3223
3224
3225
3226
3227
3228
3229
3230
3231
3232
3233
3234
3235
3236
3237
3238
3239
3240
3241
3242
3243
3244
3245
3246
3247
3248
3249
3250
3251
3252
3253
3254
3255
3256
3257
3258
3259
3260
3261
3262
3263
3264
3265
3266
3267
3268
3269
3270
3271
3272
3273
3274
3275
3276
3277
3278
3279
3280
3281
3282
3283
3284
3285
3286
3287
3288
3289
3290
3291
3292
3293
3294
3295
3296
3297
3298
3299
3300
3301
3302
3303
3304
3305
3306
3307
3308
3309
3310
3311
3312
3313
3314
3315
3316
3317
3318
3319
3320
3321
3322
3323
3324
3325
3326
3327
3328
3329
3330
3331
3332
3333
3334
3335
3336
3337
3338
3339
3340
3341
3342
3343
3344
3345
3346
3347
3348
3349
3350
3351
3352
3353
3354
3355
3356
3357
3358
3359
3360
3361
3362
3363
3364
3365
3366
3367
3368
3369
3370
3371
3372
3373
3374
3375
3376
3377
3378
3379
3380
3381
3382
3383
3384
3385
3386
3387
3388
3389
3390
3391
3392
3393
3394
3395
3396
3397
3398
3399
3400
3401
3402
3403
3404
3405
3406
3407
3408
3409
3410
3411
3412
3413
3414
3415
3416
3417
3418
3419
3420
3421
3422
3423
3424
3425
3426
3427
3428
3429
3430
3431
3432
3433
3434
3435
3436
3437
3438
3439
3440
3441
3442
3443
3444
3445
3446
3447
3448
3449
3450
3451
3452
3453
3454
3455
3456
3457
3458
3459
3460
3461
3462
3463
3464
3465
3466
3467
3468
3469
3470
3471
3472
3473
3474
3475
3476
3477
3478
3479
3480
3481
3482
3483
3484
3485
3486
3487
3488
3489
3490
3491
3492
3493
3494
3495
3496
3497
3498
3499
3500
3501
3502
3503
3504
3505
3506
3507
3508
3509
3510
3511
3512
3513
3514
3515
3516
3517
3518
3519
3520
3521
3522
3523
3524
3525
3526
3527
3528
3529
3530
3531
3532
3533
3534
3535
3536
3537
3538
3539
3540
3541
3542
3543
3544
3545
3546
3547
3548
3549
3550
3551
3552
3553
3554
3555
3556
3557
3558
3559
3560
3561
3562
3563
3564
3565
3566
3567
3568
3569
3570
3571
3572
3573
3574
3575
3576
3577
3578
3579
3580
3581
3582
3583
3584
3585
3586
3587
3588
3589
3590
3591
3592
3593
3594
3595
3596
3597
3598
3599
3600
3601
3602
3603
3604
3605
3606
3607
3608
3609
3610
3611
3612
3613
3614
3615
3616
3617
3618
3619
3620
3621
3622
3623
3624
3625
3626
3627
3628
3629
3630
3631
3632
3633
3634
3635
3636
3637
3638
3639
3640
3641
3642
3643
3644
3645
3646
3647
3648
3649
3650
3651
3652
3653
3654
3655
3656
3657
3658
3659
3660
3661
3662
3663
3664
3665
3666
3667
3668
3669
3670
3671
3672
3673
3674
3675
3676
3677
3678
3679
3680
3681
3682
3683
3684
3685
3686
3687
3688
3689
3690
3691
3692
3693
3694
3695
3696
3697
3698
3699
3700
3701
3702
3703
3704
3705
3706
3707
3708
3709
3710
3711
3712
3713
3714
3715
3716
3717
3718
3719
3720
3721
3722
3723
3724
3725
3726
3727
3728
3729
3730
3731
3732
3733
3734
3735
3736
3737
3738
3739
3740
3741
3742
3743
3744
3745
3746
3747
3748
3749
3750
3751
3752
3753
3754
3755
3756
3757
3758
3759
3760
3761
3762
3763
3764
3765
3766
3767
3768
3769
3770
3771
3772
3773
3774
3775
3776
3777
3778
3779
3780
3781
3782
3783
3784
3785
3786
3787
3788
3789
3790
3791
3792
3793
3794
3795
3796
3797
3798
3799
3800
3801
3802
3803
3804
3805
3806
3807
3808
3809
3810
3811
3812
3813
3814
3815
3816
3817
3818
3819
3820
3821
3822
3823
3824
3825
3826
3827
3828
3829
3830
3831
3832
3833
3834
3835
3836
3837
3838
3839
3840
3841
3842
3843
3844
3845
3846
3847
3848
3849
3850
3851
3852
3853
3854
3855
3856
3857
3858
3859
3860
3861
3862
3863
3864
3865
3866
3867
3868
3869
3870
3871
3872
3873
3874
3875
3876
3877
3878
3879
3880
3881
3882
3883
3884
3885
3886
3887
3888
3889
3890
3891
3892
3893
3894
3895
3896
3897
3898
3899
3900
3901
3902
3903
3904
3905
3906
3907
3908
3909
3910
3911
3912
3913
3914
3915
3916
3917
3918
3919
3920
3921
3922
3923
3924
3925
3926
3927
3928
3929
3930
3931
3932
3933
3934
3935
3936
3937
3938
3939
3940
3941
3942
3943
3944
3945
3946
3947
3948
3949
3950
3951
3952
3953
3954
3955
3956
3957
3958
3959
3960
3961
3962
3963
3964
3965
3966
3967
3968
3969
3970
3971
3972
3973
3974
3975
3976
3977
3978
3979
3980
3981
3982
3983
3984
3985
3986
3987
3988
3989
3990
3991
3992
3993
3994
3995
3996
3997
3998
3999
4000
4001
4002
4003
4004
4005
4006
4007
4008
4009
4010
4011
4012
4013
4014
4015
4016
4017
4018
4019
4020
4021
4022
4023
4024
4025
4026
4027
4028
4029
4030
4031
4032
4033
4034
4035
4036
4037
4038
4039
4040
4041
4042
4043
4044
4045
4046
4047
4048
4049
4050
4051
4052
4053
4054
4055
4056
4057
4058
4059
4060
4061
4062
4063
4064
4065
4066
4067
4068
4069
4070
4071
4072
4073
4074
4075
4076
4077
4078
4079
4080
4081
4082
4083
4084
4085
4086
4087
4088
4089
4090
4091
4092
4093
4094
4095
4096
4097
4098
4099
4100
4101
4102
4103
4104
4105
4106
4107
4108
4109
4110
4111
4112
4113
4114
4115
4116
4117
4118
4119
4120
4121
4122
4123
4124
4125
4126
4127
4128
4129
4130
4131
4132
4133
4134
4135
4136
4137
4138
4139
4140
4141
4142
4143
4144
4145
4146
4147
4148
4149
4150
4151
4152
4153
4154
4155
4156
4157
4158
4159
4160
4161
4162
4163
4164
4165
4166
4167
4168
4169
4170
4171
4172
4173
4174
4175
4176
4177
4178
4179
4180
4181
4182
4183
4184
4185
4186
4187
4188
4189
4190
4191
4192
4193
4194
4195
4196
4197
4198
4199
4200
4201
4202
4203
4204
4205
4206
4207
4208
4209
4210
4211
4212
4213
4214
4215
4216
4217
4218
4219
4220
4221
4222
4223
4224
4225
4226
4227
4228
4229
4230
4231
4232
4233
4234
4235
4236
4237
4238
4239
4240
4241
4242
4243
4244
4245
4246
4247
4248
4249
4250
4251
4252
4253
4254
4255
4256
4257
4258
4259
4260
4261
4262
4263
4264
4265
4266
4267
4268
4269
4270
4271
4272
4273
4274
4275
4276
4277
4278
4279
4280
4281
4282
4283
4284
4285
4286
4287
4288
4289
4290
4291
4292
4293
4294
4295
4296
4297
4298
4299
4300
4301
4302
4303
4304
4305
4
```

```

2322     11)) = (11 : ℝ) ^ (2 * (log (11 / 5) / log 11)) = (11 : ℝ) ^ (log
2323     (11 / 5) / (log 11 / 2)) = exp (log (11 / 5) / (log 11 / 2) * log
2324     11) = exp (2 * log (11 / 5)) = (11 / 5) ^ 2 = 121 / 25'.
2325
2326 98  99 This is correct because '(11 : ℝ) ^ (1 / 4 : ℝ) > 0', and 'log
2327     ((11 : ℝ) ^ (1 / 4 : ℝ)) = (1 / 4) * log 11'.
2328
2329 100 101 ### Step 1: Abstract Plan
2330
2331 102 1. Find 'x' in terms of logarithms**:
2332     - Take the natural logarithm of both sides of 'h' to find 'x'
2333     in terms of 'log 5' and 'log 11'.
2334
2335 103 2. Substitute 'x' into the goal**:
2336     - Substitute 'x = 1 - (4 / 3) * (log 5 / log 11)' into '6 * x +
2337     2' to find '6 * x + 2 = 8 - 8 * (log 5 / log 11)'.
2338
2339 104 3. Simplify the exponent**:
2340     - Rewrite '((11 : ℝ) ^ (1 / 4 : ℝ)) ^ (6 * x + 2)' as '((11 : ℝ
2341     ) ^ (1 / 4 : ℝ)) ^ (8 * (1 - (log 5 / log 11)))'.
2342
2343 105 111 4. Simplify using logarithms**:
2344     - Use the logarithm power rule to find '((11 : ℝ) ^ (1 / 4 : ℝ
2345     )) ^ (8 * (1 - (log 5 / log 11))) = 121 / 25'.
2346
2347 112 114 ### Step 2: Lean 4 'have' Statements
2348
2349 115 116 ```lean4
2350 116 117 theorem mathd_algebra_275 (x : ℝ) (h : ((11 : ℝ) ^ (1 / 4 : ℝ)) ^
2351 117     (3 * x - 3) = 1 / 5) :
2352 118     ((11 : ℝ) ^ (1 / 4 : ℝ)) ^ (6 * x + 2) = 121 / 25 := by
2353 119     have h_main : ((11 : ℝ) ^ (1 / 4 : ℝ)) ^ (6 * x + 2) = 121 / 25
2354 120     := by
2355 121     sorry
2356 122     sorry
2357 123     ```
2358
2359 124
2360 125 ### Complete Lean 4 Proof
2361
2362 126 ```lean4
2363 127 theorem mathd_algebra_275 (x : ℝ) (h : ((11 : ℝ) ^ (1 / 4 : ℝ)) ^
2364 127     (3 * x - 3) = 1 / 5) :
2365 128     ((11 : ℝ) ^ (1 / 4 : ℝ)) ^ (6 * x + 2) = 121 / 25 := by
2366 129     have h_main : ((11 : ℝ) ^ (1 / 4 : ℝ)) ^ (6 * x + 2) = 121 / 25
2367 130     := by
2368     have h1 : Real.log (((11 : ℝ) ^ (1 / 4 : ℝ)) ^ (3 * x - 3)) =
2369     Real.log (1 / 5) := by rw [h]
2370     have h2 : (3 * x - 3) * Real.log ((11 : ℝ) ^ (1 / 4 : ℝ)) =
2371     Real.log (1 / 5) := by
2372     have h3 : Real.log (((11 : ℝ) ^ (1 / 4 : ℝ)) ^ (3 * x - 3)) =
2373     (3 * x - 3) * Real.log ((11 : ℝ) ^ (1 / 4 : ℝ)) := by
2374     rw [Real.log_rpow (by positivity)]
2375     < ; > ring_nf
2376     < ; > field_simp
2377     < ; > ring_nf
2378     rw [h3] at h1
2379     linarith
2380
2381     have h3 : (3 * x - 3) * Real.log ((11 : ℝ) ^ (1 / 4 : ℝ)) =
2382     Real.log (1 / 5) := by linarith
2383     have h4 : Real.log ((11 : ℝ) ^ (1 / 4 : ℝ)) = (1 / 4 : ℝ) *
2384     Real.log 11 := by
2385     rw [Real.log_rpow (by positivity)]
2386     < ; > ring_nf
2387     < ; > field_simp
2388     < ; > ring_nf

```

```

2376      rw [h4] at h3
2377      have h5 : (3 * x - 3) * ((1 / 4 : ℝ) * Real.log 11) = Real.log
2378      (1 / 5) := by linarith
2379      have h6 : (3 * x - 3) * ((1 / 4 : ℝ) * Real.log 11) = Real.log
2380      (1 / 5) := by linarith
2381      have h7 : Real.log (1 / 5) = - Real.log 5 := by
2382      have h71 : Real.log (1 / 5) = Real.log 1 - Real.log 5 := by
2383      rw [Real.log_div (by norm_num) (by norm_num)]
2384      <;> simp
2385      rw [h71]
2386      have h72 : Real.log 1 = 0 := by norm_num
2387      rw [h72]
2388      <;> ring
2389      <;> linarith
2390      rw [h7] at h6
2391      have h8 : (3 * x - 3) * ((1 / 4 : ℝ) * Real.log 11) = -
2392      Real.log 5 := by linarith
2393      have h9 : x = 1 - (4 / 3) * (Real.log 5 / Real.log 11) := by
2394      have h91 : (3 * x - 3) * ((1 / 4 : ℝ) * Real.log 11) = -
2395      Real.log 5 := by linarith
2396      have h92 : 3 * x - 3 = - (4 : ℝ) * (Real.log 5 / Real.log
2397      11) := by
2398      have h93 : Real.log 11 ≠ 0 := by
2399      exact Real.log_ne_zero_of_pos_of_ne_one (by norm_num)
2400      (by norm_num)
2401      field_simp at h91 +
2402      nlinarith
2403      have h93 : x = 1 - (4 / 3) * (Real.log 5 / Real.log 11) := by
2404      have h94 : 3 * x - 3 = - (4 : ℝ) * (Real.log 5 / Real.log
2405      11) := by linarith
2406      have h95 : x = 1 - (4 / 3) * (Real.log 5 / Real.log 11) := by
2407      by
2408      have h96 : 3 * x - 3 = - (4 : ℝ) * (Real.log 5 /
2409      Real.log 11) := by linarith
2410      have h97 : x = 1 - (4 / 3) * (Real.log 5 / Real.log 11) := by
2411      by
2412      apply Eq.symm
2413      field_simp at h96 +
2414      nlinarith
2415      exact h97
2416      exact h95
2417      exact h93
2418      have h10 : ((11 : ℝ) ^ (1 / 4 : ℝ)) ^ (6 * x + 2) = 121 / 25 :=
2419      by
2420      rw [h9]
2421      have h11 : ((11 : ℝ) ^ (1 / 4 : ℝ)) ^ (6 * (1 - (4 / 3) *
2422      (Real.log 5 / Real.log 11)) + 2) = 121 / 25 := by
2423      have h12 : ((11 : ℝ) ^ (1 / 4 : ℝ)) ^ (6 * (1 - (4 / 3) *
2424      (Real.log 5 / Real.log 11)) + 2) = ((11 : ℝ) ^ (1 / 4 : ℝ)) ^ (8 *
2425      (1 - (Real.log 5 / Real.log 11))) := by
2426      ring_nf
2427      <;> field_simp
2428      <;> ring_nf
2429      <;> norm_num
2430      <;> linarith
2431      rw [h12]
2432      have h13 : ((11 : ℝ) ^ (1 / 4 : ℝ)) ^ (8 * (1 - (Real.log
2433      5 / Real.log 11))) = 121 / 25 := by
2434      have h14 : Real.log (((11 : ℝ) ^ (1 / 4 : ℝ)) ^ (8 * (1
2435      - (Real.log 5 / Real.log 11)))) = Real.log (121 / 25) := by
2436      have h15 : Real.log (((11 : ℝ) ^ (1 / 4 : ℝ)) ^ (8 *
2437      (1 - (Real.log 5 / Real.log 11)))) = (8 * (1 - (Real.log 5 /
2438      Real.log 11))) * Real.log ((11 : ℝ) ^ (1 / 4 : ℝ)) := by
2439      rw [Real.log_rpow (by positivity)]
2440      <;> ring_nf

```

```

2430
2431
2432
2433
2434
2435
2436
2437
2438
2439
2440
2441
2442
2443
2444
2445
2446
2447
2448
2449
2450
2451
2452
2453
2454
2455
2456
2457
2458
2459
2460
2461
2462
2463
2464
2465
2466
2467
2468
2469
2470
2471
2472
2473
2474
2475
2476
2477
2478
2479
2480
2481
2482
2483
193      <;> field_simp
194      <;> ring_nf
195      rw [h15]
196      have h16 : Real.log ((11 : ℝ) ^ (1 / 4 : ℝ)) = (1 / 4
: ℝ) * Real.log 11 := by
197      rw [Real.log_rpow (by positivity)]
198      <;> ring_nf
199      <;> field_simp
200      <;> ring_nf
201      rw [h16]
202      have h17 : Real.log (121 / 25) = Real.log (121 / 25) :=
203      by rfl
204      have h18 : (8 * (1 - (Real.log 5 / Real.log 11))) * ((1
/ 4 : ℝ) * Real.log 11) = Real.log (121 / 25) := by
205      have h19 : Real.log (121 / 25) = Real.log (121 / 25)
206      := by rfl
207      have h20 : Real.log 121 = Real.log (11 ^ 2) := by
208      norm_num
209      have h21 : Real.log (11 ^ 2) = 2 * Real.log 11 := by
210      rw [Real.log_pow] <;> norm_num
211      have h22 : Real.log 25 = Real.log (5 ^ 2) := by
212      norm_num
213      have h23 : Real.log (5 ^ 2) = 2 * Real.log 5 := by
214      rw [Real.log_pow] <;> norm_num
215      have h24 : Real.log (121 / 25) = Real.log 121 -
216      Real.log 25 := by
217      rw [Real.log_div (by positivity) (by positivity)]
218      <;> ring_nf
219      <;> field_simp
220      <;> ring_nf
221      <;> ring_nf
222      <;> nlinarith
223      linarith
224      linarith
225      have h15 : Real.log (((11 : ℝ) ^ (1 / 4 : ℝ)) ^ (8 * (1
- (Real.log 5 / Real.log 11))) = Real.log (121 / 25) := by
226      exact h14
227      have h16 : ((11 : ℝ) ^ (1 / 4 : ℝ)) ^ (8 * (1 -
(Real.log 5 / Real.log 11))) > 0 := by positivity
228      have h17 : Real.log (((11 : ℝ) ^ (1 / 4 : ℝ)) ^ (8 * (1
- (Real.log 5 / Real.log 11))) = Real.log (121 / 25) := by
229      exact h15
230      have h18 : ((11 : ℝ) ^ (1 / 4 : ℝ)) ^ (8 * (1 -
(Real.log 5 / Real.log 11))) = 121 / 25 := by
231      apply Real.log_injOn_pos (Set.mem_Ioi.mpr (by
positivity)) (Set.mem_Ioi.mpr (by positivity))
232      linarith
233      exact h18
234      rw [h13]
235      <;> norm_num
236      <;> linarith
237      exact h11
238      exact h10
239      exact h_main
240      ```

```

2484
2485

E.2.5 COMPARISON: DIRECT PROOF ATTEMPT WITHOUT ITERATIVE REFINEMENT

2486
2487
2488
2489

For comparison, we present below the reasoning process and generated Lean code from a direct proof attempt without using auxiliary lemmas. Among the N_{init} initial direct proof attempts, we show the one that resulted in the fewest Lean errors and used as a initial draft for the iterative refinement process.

2490

2491

2492

2493

2494

2495

2496

2497

2498

2499

2500

2501

2502

2503

2504

2505

2506

2507

2508

2509

2510

2511

2512

2513

2514

2515

2516

2517

2518

2519

2520

2521

2522

2523

2524

2525

2526

2527

2528

2529

2530

2531

2532

2533

2534

2535

2536

2537

```

1  #### Detailed Proof and Analysis
2
3  First, let's recall the problem:
4  We have  $\langle x \in \mathbb{R} \rangle$  and the equation:
5   $\left[ \left(11^{\frac{1}{4}}\right)^{3x-3} = \frac{1}{5} \right].$ 
6  We need to prove that:
7   $\left[ \left(11^{\frac{1}{4}}\right)^{6x+2} = \frac{121}{25} \right].$ 
8
9  Let  $\langle a = 11^{\frac{1}{4}} \rangle$ . Then the given equation is:
10  $\left[ a^{3x-3} = \frac{1}{5} \right].$ 
11 The goal is to find  $\langle a^{6x+2} \rangle$ .
12
13 ##### Step 1: Express  $\langle 6x+2 \rangle$  in terms of  $\langle 3x-3 \rangle$ 
14 Notice that:
15  $\left[ 6x+2 = 2(3x-3) + 8 \right]$ 
16 This is because:
17  $\left[ 2(3x-3) + 8 = 6x - 6 + 8 = 6x + 2 \right]$ 
18
19 ##### Step 2: Rewrite the Goal Using the Exponent Identity
20 Using the exponent rule  $\langle (a^m)^n = a^{m \cdot n} \rangle$ , we have:
21  $\left[ a^{6x+2} = a^{2(3x-3) + 8} = (a^{3x-3})^2 \cdot a^8 \right]$ 
22 Substituting the given value  $\langle a^{3x-3} = \frac{1}{5} \rangle$ , we get:
23  $\left[ a^{6x+2} = \left(\frac{1}{5}\right)^2 \cdot a^8 = \frac{1}{25} \cdot a^8 \right]$ 
24
25 ##### Step 3: Compute  $\langle a^8 \rangle$ 
26 Since  $\langle a = 11^{\frac{1}{4}} \rangle$ , we have:
27  $\left[ a^4 = 11 \right]$ 
28 Therefore:
29  $\left[ a^8 = (a^4)^2 = 11^2 = 121 \right]$ 
30 Thus:
31  $\left[ a^{6x+2} = \frac{1}{25} \cdot 121 = \frac{121}{25} \right]$ 
32
33 This completes the proof.
34
35 #### Step-by-Step Abstract Plan
36
37 1. **Define  $\langle a \rangle$ **:
38  - Let  $\langle a = 11^{\frac{1}{4}} \rangle$ .
39
40 2. **Rewrite the Given Equation**:
41  - The given equation is  $\langle a^{3x-3} = \frac{1}{5} \rangle$ .
42
43 3. **Express the Goal Exponent**:
44  - Express  $\langle 6x+2 \rangle$  as  $\langle 2(3x-3) + 8 \rangle$ .
45
46 4. **Use Exponent Rules**:
47  - Rewrite  $\langle a^{6x+2} \rangle$  as  $\langle (a^{3x-3})^2 \cdot a^8 \rangle$ .
48
49 5. **Substitute the Given Value**:
50  - Substitute  $\langle a^{3x-3} = \frac{1}{5} \rangle$  to get  $\langle \left(\frac{1}{5}\right)^2 \cdot a^8 \rangle$ .
51
52 6. **Compute  $\langle a^8 \rangle$ **:
53  - Since  $\langle a^4 = 11 \rangle$ , we have  $\langle a^8 = 121 \rangle$ .
54
55 7. **Final Calculation**:

```

```

2538      - Multiply to get  $\frac{121}{25}$ .
2539
2540 56  ````lean4
2541 57  theorem mathd_algebra_275 (x : ℝ) (h : ((11 : ℝ) ^ (1 / 4 : ℝ)) ^ (3 * x - 3) = 1 / 5) :
2542 58  ((11 : ℝ) ^ (1 / 4 : ℝ)) ^ (6 * x + 2) = 121 / 25 := by
2543 59  have h_main : ((11 : ℝ) ^ (1 / 4 : ℝ)) ^ (6 * x + 2) = 121 / 25
2544 60  := by
2545 61  sorry
2546 62  sorry
2547 63  ```
2548 64
2549 65
2550 66
2551 67
2552 68  ```Explanation of 'have' Statements
2553 69
2554 70 1. **'h_main'**: This is the main goal of the proof. The proof
2555 71 will directly use the steps outlined above to derive the result.
2556 72 The actual Lean 4 proof would fill in the 'sorry' by carefully
2557 73 applying exponent rules and arithmetic simplifications.
2558 74  ```Complete Lean 4 Proof
2559 75 theorem mathd_algebra_275 (x : ℝ) (h : ((11 : ℝ) ^ (1 / 4 : ℝ)) ^ (3 * x - 3) = 1 / 5) :
2560 76  ((11 : ℝ) ^ (1 / 4 : ℝ)) ^ (6 * x + 2) = 121 / 25 := by
2561 77  have h_main : ((11 : ℝ) ^ (1 / 4 : ℝ)) ^ (6 * x + 2) = 121 / 25
2562 78  := by
2563 79  have h1 : ((11 : ℝ) ^ (1 / 4 : ℝ)) ^ (6 * x + 2) = ((11 : ℝ)
2564 80  ^ (1 / 4 : ℝ)) ^ (3 * x - 3) ^ 2 * ((11 : ℝ) ^ (1 / 4 : ℝ)) ^ 8 := by
2565 81  ring_nf
2566 82  <;>
2567 83  field_simp
2568 84  <;>
2569 85  ring_nf
2570 86  <;>
2571 87  norm_num
2572 88  <;>
2573 89  linarith
2574 90  rw [h2]
2575 91  have h3 : ((11 : ℝ) ^ (1 / 4 : ℝ)) ^ (2 * (3 * x - 3) + 8) =
2576 92  ((11 : ℝ) ^ (1 / 4 : ℝ)) ^ (3 * x - 3) ^ 2 * ((11 : ℝ) ^ (1 / 4
2577 93  : ℝ)) ^ 8 := by
2578 94  have h4 : ((11 : ℝ) ^ (1 / 4 : ℝ)) ^ (2 * (3 * x - 3) + 8) =
2579 95  = ((11 : ℝ) ^ (1 / 4 : ℝ)) ^ (2 * (3 * x - 3)) * ((11 : ℝ) ^ (1 /
2580 96  4 : ℝ)) ^ 8 := by
2581 97  rw [Real.rpow_add (by positivity)]
2582 98  <;> ring_nf
2583 99  rw [h4]
2584 100  have h5 : ((11 : ℝ) ^ (1 / 4 : ℝ)) ^ (2 * (3 * x - 3)) =
2585 101  ((11 : ℝ) ^ (1 / 4 : ℝ)) ^ (3 * x - 3) ^ 2 := by
2586 102  have h6 : ((11 : ℝ) ^ (1 / 4 : ℝ)) ^ (2 * (3 * x - 3)) =
2587 103  ((11 : ℝ) ^ (1 / 4 : ℝ)) ^ (3 * x - 3) ^ 2 := by
2588 104  rw [show 2 * (3 * x - 3) = (3 * x - 3) + (3 * x - 3)
2589 105  by ring]
2590 106  rw [show ((11 : ℝ) ^ (1 / 4 : ℝ)) ^ ((3 * x - 3) + (3 *
2591 107  x - 3)) = ((11 : ℝ) ^ (1 / 4 : ℝ)) ^ (3 * x - 3) * ((11 : ℝ) ^ (1
2592 108  / 4 : ℝ)) ^ (3 * x - 3) by
2593 109  rw [Real.rpow_add (by positivity)] <;> ring_nf
2594 110  <;> ring_nf
2595 111  <;>

```

```

2592           field_simp
2593           <;>
2594           ring_nf
2595           <;>
2596           norm_num
2597           <;>
2598           linarith
2599           rw [h6]
2600           rw [h5]
2601           <;> ring_nf
2602           <;>
2603           field_simp
2604           <;>
2605           ring_nf
2606           <;>
2607           norm_num
2608           <;>
2609           linarith
2610           rw [h3]
2611           <;> ring_nf
2612           <;>
2613           field_simp
2614           <;>
2615           ring_nf
2616           <;>
2617           norm_num
2618           <;>
2619           linarith
2620           rw [h1]
2621           have h2 : (((11 : ℝ) ^ (1 / 4 : ℝ)) ^ (3 * x - 3)) ^ 2 = (1 /
2622           5) ^ 2 := by
2623             rw [h]
2624             <;> ring_nf
2625             <;> norm_num
2626             rw [h2]
2627             have h3 : ((11 : ℝ) ^ (1 / 4 : ℝ)) ^ 8 = 121 := by
2628               have h4 : ((11 : ℝ) ^ (1 / 4 : ℝ)) ^ 8 = 121 := by
2629               have h5 : ((11 : ℝ) ^ (1 / 4 : ℝ)) ^ 8 = (11 : ℝ) ^ ((1 /
2630               4 : ℝ) * 8) := by
2631                 rw [← Real.rpow_nat_cast]
2632                 rw [← Real.rpow_mul] <;> ring_nf <;> norm_num
2633                 <;> positivity
2634                 rw [h5]
2635                 have h6 : (11 : ℝ) ^ ((1 / 4 : ℝ) * 8) = (11 : ℝ) ^ (2 : ℝ)
2636               ) := by
2637                 norm_num
2638                 <;> ring_nf
2639                 <;> field_simp
2640                 <;> ring_nf
2641                 rw [h6]
2642                 have h7 : (11 : ℝ) ^ (2 : ℝ) = 121 := by
2643                   norm_num [Real.rpow_two]
2644                   <;> ring_nf
2645                   <;> norm_num
2646                   rw [h7]
2647                   <;> norm_num
2648                   rw [h4]
2649                   rw [h3]
2650                   <;> norm_num
2651                   <;> ring_nf
2652                   <;> field_simp
2653                   <;> ring_nf
2654                   <;> norm_num
2655                   <;> linarith
2656                   exact h_main

```

2646 164 ```

2647

2648

2649 F PROMPTS USED IN OUR EXPERIMENTS

2650

2651 In this section, we provide the prompts used in our experiments for the various stages of the Prover
2652 Agent pipeline.

2653

2654 F.1 THE PROMPT FOR INITIAL DIRECT PROVING

2655

2656 The prompt provided to the informal LLM at the initial direct proving stage is as follows:

2657

```

1 Your goal is to implement the following theorem, using Lean 4 and
2 the mathlib library:
3 ```lean4
4 {lean_header}
5
6
7 {theorem}
8 ```
9
10 First, provide a step-by-step proof in English.
11 DO NOT write Lean code here yet--just write the proof in English.

```

2668

2669 F.2 THE PROMPT FOR INITIAL DIRECT PROVING

2670

2671 The prompt provided to the prove model at the initial direct proving stage is as follows:

2672

```

1 Your goal is to implement the following theorem, using Lean 4 and
2 the mathlib library:
3 ```lean4
4 {lean_header}
5
6
7 {theorem}
8 ```
9
10 The English proof is as follows:
11
12 ```text
13 {nl_proof}
14 ```
15
16 Complete the following Lean 4 code:
17
18 ```lean4
19 {lean_header}
20
21
22 {theorem}
23 ```
24
25 Before producing the Lean 4 code to formally prove the given
26 theorem, provide a detailed proof plan outlining the main proof
27 steps and strategies.
28 The plan should highlight key ideas, intermediate lemmas, and
29 proof structures that will guide the construction of the final
30 formal proof.

```

2699

Here, “nl_proof” is the output from the informal LLM at the initial direct proving stage.

2700
2701

F.3 THE PROMPT FOR ITERATIVE REFINEMENT IN DIRECT PROVING

2702
2703

The prompt for the iterative refinement stage in direct proving is as follows:

2704
2705

```

1 Your goal is to implement the following theorem, using Lean 4 and
2 the mathlib library:
3 ````lean4
4 {lean_header}
5
6
7 {theorem}
8 ``
9
10 Your proof is as follows:
11
12 ````lean4
13 {prev_code}
14 ``
15
16 The proof failed to compile with errors.
17 The error occurred at the following line(s):
18
19 {error_line_messages}
20
21 Fix these errors and complete the following Lean 4 code:
22
23 ````lean4
24 {lean_header}
25
26
27 {theorem}
28 ``
29
30 Before producing the Lean 4 code to formally prove the given
31 theorem, provide a detailed proof plan outlining the main proof
32 steps and strategies.
33 The plan should highlight key ideas, intermediate lemmas, and
34 proof structures that will guide the construction of the final
35 formal proof.
36

```

2735
2736

Here, the “prev_code” is the previous Lean code generated by the prove model. The “error_line_messages” is formatted as follows, and this block is repeated for every error:

2737

```

1 Error line (line {error_line}): 
2 ````lean4
3 {error_code}
4 ``
5 Error message:
6 ````lean4
7 {error_message}
8 ``

```

2745

F.4 THE PROMPT FOR LEMMA GENERATION

2746
2747
2748

The prompt provided to the informal LLM for lemma generation is as follows:

2749

```

1 I am trying to code (prove) the following theorem in Lean 4.
2
3 ````lean4
4 {lean_header}
5
6

```

```

2754 7  {theorem}
2755 8  ``
2756 9
2757 10 Derive {num_lemmas} lemmas related to the theorem.
2758 11 The related lemmas are those that could serve as subpropositions,
2759 12 For example, consider treating the case where a specific value is
2760 13 For each lemma, clearly state the assumptions and the conclusion
2761 14 Include any assumptions from the original theorem as needed in
2762 15 You do not need to write the proofs or the Lean code for each
2763 16 Follow the format below for each lemma:
2764 17 ``
2765 18 ``
2766 19 ### Lemma 1: <Lemma Name>
2767 20 **Assumptions**:
2768 21 <Assumptions in English>
2769 22
2770 23 **Conclusion**:
2771 24 <Conclusion in English>
2772 25 ``
2773 26 Do not include any explanations or additional text outside of the
2774 27 specified format.
2775
2776
2777

```

2778 Here, “num_lemmas” is set to 3 in our experiments.

2781 F.5 THE PROMPT FOR LEMMA FORMALIZATION

2782 The prompt provided to the formalizer model for lemma formalization is as follows:

```

2783 1 Please autoformalize the following natural language problem
2784 2 statement in Lean 4. Use the following theorem name: {problem_name}
2785 3 The natural language statement is:
2786 4 {nl_statement}
2787 5 Think before you provide the lean statement.
2788
2789
2790

```

2791 Here, “problem_name” is the name of the lemma taken directly from the <Lemma Name> field in
2792 the output of the lemma generation step.

2794 F.6 THE PROMPT FOR FINAL SYNTHESIS

2795 The prompt provided to the prover model at the final synthesis stage is as follows:

```

2796 1 Based on these lemmas, construct and complete the following Lean 4
2797 2 code:
2798 3 ````lean4
2799 4 {lean_header}
2800 5
2801 6
2802 7 {lemmas}
2803 8
2804 9 {theorem}
2805 10 ``
2806
2807
2808
2809
2810
2811

```

```

2808 12 Before producing the Lean 4 code to formally prove the given
2809 theorem, provide a detailed proof plan outlining the main proof
2810 steps and strategies.
2811 13 The plan should highlight key ideas, intermediate lemmas, and
2812 proof structures that will guide the construction of the final
2813 formal proof.

```

2814 Here, “lemmas” is the concatenation of the proved lemmas.

2815

2816 F.7 THE PROMPT FOR ITERATIVE REFINEMENT IN FINAL SYNTHESIS

2817

2818 The prompt provided to the prover model at the iterative refinement stage in final synthesis is as
2819 follows:

```

2820 1 Your goal is to implement the following theorem, using Lean 4 and
2821 the mathlib library:
2822 2
2823 3 ````lean4
2824 4 {lean_header}
2825 5
2826 6
2827 7 {theorem}
2828 8 ``
2829 9
2830 10 Based on lemmas, you are trying to construct the proof for the
2831 theorem.
2832 11 Your proof is as follows:
2833 12
2834 13 ````lean4
2835 14 {prev_code}
2836 15 ``
2837 16
2838 17 The proof failed to compile with errors.
2839 18 The error occurred at the following line(s):
2840 19
2841 20 {error_line_messages}
2842 21
2843 22 Fix the errors and complete the following Lean 4 code
2844 23
2845 24 ````lean4
2846 25 {lean_header}
2847 26
2848 27
2849 28 {lemmas}
2850 29
2851 30 {theorem}
2852 31 ``
2853 32
2854 33 Before producing the Lean 4 code to formally prove the given
2855 theorem, provide a detailed proof plan outlining the main proof
2856 steps and strategies.
2857 34 The plan should highlight key ideas, intermediate lemmas, and
2858 proof structures that will guide the construction of the final
2859 formal proof.

```

2860 Here, “lemmas” is the concatenation of the proved lemmas, “prev_code” is the previous Lean code
2861 generated by the prover model, and “error_line_messages” is formatted in the same way as in the
2862 iterative refinement stage in direct proving.

2863
2864
2865
2866
2867
2868
2869
2870
2871
2872
2873
2874
2875
2876
2877
2878
2879
2880
2881
2882
2883
2884
2885
2886
2887
2888
2889
2890
2891
2892
2893
2894
2895
2896
2897
2898
2899
2900
2901
2902
2903
2904
2905
2906
2907
2908
2909
2910
2911
2912
2913
2914
2915
2916
2917
2918
2919
2920
2921
2922
2923
2924
2925
2926
2927
2928
2929
2930
2931
2932
2933
2934
2935
2936
2937
2938
2939
2940
2941
2942
2943
2944
2945
2946
2947
2948
2949
2950
2951
2952
2953
2954
2955
2956
2957
2958
2959
2960
2961
2962
2963
2964
2965
2966
2967
2968
2969
2970
2971
2972
2973
2974
2975
2976
2977
2978
2979
2980
2981
2982
2983
2984
2985
2986
2987
2988
2989
2990
2991
2992
2993
2994
2995
2996
2997
2998
2999
3000
3001
3002
3003
3004
3005
3006
3007
3008
3009
3010
3011
3012
3013
3014
3015
3016
3017
3018
3019
3020
3021
3022
3023
3024
3025
3026
3027
3028
3029
3030
3031
3032
3033
3034
3035
3036
3037
3038
3039
3040
3041
3042
3043
3044
3045
3046
3047
3048
3049
3050
3051
3052
3053
3054
3055
3056
3057
3058
3059
3060
3061
3062
3063
3064
3065
3066
3067
3068
3069
3070
3071
3072
3073
3074
3075
3076
3077
3078
3079
3080
3081
3082
3083
3084
3085
3086
3087
3088
3089
3090
3091
3092
3093
3094
3095
3096
3097
3098
3099
3100
3101
3102
3103
3104
3105
3106
3107
3108
3109
3110
3111
3112
3113
3114
3115
3116
3117
3118
3119
3120
3121
3122
3123
3124
3125
3126
3127
3128
3129
3130
3131
3132
3133
3134
3135
3136
3137
3138
3139
3140
3141
3142
3143
3144
3145
3146
3147
3148
3149
3150
3151
3152
3153
3154
3155
3156
3157
3158
3159
3160
3161
3162
3163
3164
3165
3166
3167
3168
3169
3170
3171
3172
3173
3174
3175
3176
3177
3178
3179
3180
3181
3182
3183
3184
3185
3186
3187
3188
3189
3190
3191
3192
3193
3194
3195
3196
3197
3198
3199
3200
3201
3202
3203
3204
3205
3206
3207
3208
3209
3210
3211
3212
3213
3214
3215
3216
3217
3218
3219
3220
3221
3222
3223
3224
3225
3226
3227
3228
3229
3230
3231
3232
3233
3234
3235
3236
3237
3238
3239
3240
3241
3242
3243
3244
3245
3246
3247
3248
3249
3250
3251
3252
3253
3254
3255
3256
3257
3258
3259
3260
3261
3262
3263
3264
3265
3266
3267
3268
3269
3270
3271
3272
3273
3274
3275
3276
3277
3278
3279
3280
3281
3282
3283
3284
3285
3286
3287
3288
3289
3290
3291
3292
3293
3294
3295
3296
3297
3298
3299
3300
3301
3302
3303
3304
3305
3306
3307
3308
3309
3310
3311
3312
3313
3314
3315
3316
3317
3318
3319
3320
3321
3322
3323
3324
3325
3326
3327
3328
3329
3330
3331
3332
3333
3334
3335
3336
3337
3338
3339
3340
3341
3342
3343
3344
3345
3346
3347
3348
3349
3350
3351
3352
3353
3354
3355
3356
3357
3358
3359
3360
3361
3362
3363
3364
3365
3366
3367
3368
3369
3370
3371
3372
3373
3374
3375
3376
3377
3378
3379
3380
3381
3382
3383
3384
3385
3386
3387
3388
3389
3390
3391
3392
3393
3394
3395
3396
3397
3398
3399
3400
3401
3402
3403
3404
3405
3406
3407
3408
3409
3410
3411
3412
3413
3414
3415
3416
3417
3418
3419
3420
3421
3422
3423
3424
3425
3426
3427
3428
3429
3430
3431
3432
3433
3434
3435
3436
3437
3438
3439
3440
3441
3442
3443
3444
3445
3446
3447
3448
3449
3450
3451
3452
3453
3454
3455
3456
3457
3458
3459
3460
3461
3462
3463
3464
3465
3466
3467
3468
3469
3470
3471
3472
3473
3474
3475
3476
3477
3478
3479
3480
3481
3482
3483
3484
3485
3486
3487
3488
3489
3490
3491
3492
3493
3494
3495
3496
3497
3498
3499
3500
3501
3502
3503
3504
3505
3506
3507
3508
3509
3510
3511
3512
3513
3514
3515
3516
3517
3518
3519
3520
3521
3522
3523
3524
3525
3526
3527
3528
3529
3530
3531
3532
3533
3534
3535
3536
3537
3538
3539
3540
3541
3542
3543
3544
3545
3546
3547
3548
3549
3550
3551
3552
3553
3554
3555
3556
3557
3558
3559
3560
3561
3562
3563
3564
3565
3566
3567
3568
3569
3570
3571
3572
3573
3574
3575
3576
3577
3578
3579
3580
3581
3582
3583
3584
3585
3586
3587
3588
3589
3590
3591
3592
3593
3594
3595
3596
3597
3598
3599
3600
3601
3602
3603
3604
3605
3606
3607
3608
3609
3610
3611
3612
3613
3614
3615
3616
3617
3618
3619
3620
3621
3622
3623
3624
3625
3626
3627
3628
3629
3630
3631
3632
3633
3634
3635
3636
3637
3638
3639
3640
3641
3642
3643
3644
3645
3646
3647
3648
3649
3650
3651
3652
3653
3654
3655
3656
3657
3658
3659
3660
3661
3662
3663
3664
3665
3666
3667
3668
3669
3670
3671
3672
3673
3674
3675
3676
3677
3678
3679
3680
3681
3682
3683
3684
3685
3686
3687
3688
3689
3690
3691
3692
3693
3694
3695
3696
3697
3698
3699
3700
3701
3702
3703
3704
3705
3706
3707
3708
3709
3710
3711
3712
3713
3714
3715
3716
3717
3718
3719
3720
3721
3722
3723
3724
3725
3726
3727
3728
3729
3730
3731
3732
3733
3734
3735
3736
3737
3738
3739
3740
3741
3742
3743
3744
3745
3746
3747
3748
3749
3750
3751
3752
3753
3754
3755
3756
3757
3758
3759
3760
3761
3762
3763
3764
3765
3766
3767
3768
3769
3770
3771
3772
3773
3774
3775
3776
3777
3778
3779
3780
3781
3782
3783
3784
3785
3786
3787
3788
3789
3790
3791
3792
3793
3794
3795
3796
3797
3798
3799
3800
3801
3802
3803
3804
3805
3806
3807
3808
3809
3810
3811
3812
3813
3814
3815
3816
3817
3818
3819
3820
3821
3822
3823
3824
3825
3826
3827
3828
3829
3830
3831
3832
3833
3834
3835
3836
3837
3838
3839
3840
3841
3842
3843
3844
3845
3846
3847
3848
3849
3850
3851
3852
3853
3854
3855
3856
3857
3858
3859
3860
3861
3862
3863
3864
3865
3866
3867
3868
3869
3870
3871
3872
3873
3874
3875
3876
3877
3878
3879
3880
3881
3882
3883
3884
3885
3886
3887
3888
3889
3890
3891
3892
3893
3894
3895
3896
3897
3898
3899
3900
3901
3902
3903
3904
3905
3906
3907
3908
3909
3910
3911
3912
3913
3914
3915
3916
3917
3918
3919
3920
3921
3922
3923
3924
3925
3926
3927
3928
3929
3930
3931
3932
3933
3934
3935
3936
3937
3938
3939
3940
3941
3942
3943
3944
3945
3946
3947
3948
3949
3950
3951
3952
3953
3954
3955
3956
3957
3958
3959
3960
3961
3962
3963
3964
3965
3966
3967
3968
3969
3970
3971
3972
3973
3974
3975
3976
3977
3978
3979
3980
3981
3982
3983
3984
3985
3986
3987
3988
3989
3990
3991
3992
3993
3994
3995
3996
3997
3998
3999
4000
4001
4002
4003
4004
4005
4006
4007
4008
4009
4010
4011
4012
4013
4014
4015
4016
4017
4018
4019
4020
4021
4022
4023
4024
4025
4026
4027
4028
4029
4030
4031
4032
4033
4034
4035
4036
4037
4038
4039
4040
4041
4042
4043
4044
4045
4046
4047
4048
4049
4050
4051
4052
4053
4054
4055
4056
4057
4058
4059
4060
4061
4062
4063
4064
4065
4066
4067
4068
4069
4070
4071
4072
4073
4074
4075
4076
4077
4078
4079
4080
4081
4082
4083
4084
4085
4086
4087
4088
4089
4090
4091
4092
4093
4094
4095
4096
4097
4098
4099
4100
4101
4102
4103
4104
4105
4106
4107
4108
4109
4110
4111
4112
4113
4114
4115
4116
4117
4118
4119
4120
4121
4122
4123
4124
4125
4126
4127
4128
4129
4130
4131
4132
4133
4134
4135
4136
4137
4138
4139
4140
4141
4142
4143
4144
4145
4146
4147
4148
4149
4150
4151
4152
4153
4154
4155
4156
4157
4158
4159
4160
4161
4162
4163
4164
4165
4166
4167
4168
4169
4170
4171
4172
4173
4174
4175
4176
4177
4178
4179
4180
4181
4182
4183
4184
4185
4186
4187
4188
4189
4190
4191
4192
4193
4194
4195
4196
4197
4198
4199
4200
4201
4202
4203
4204
4205
4206
4207
4208
4209
4210
4211
4212
4213
4214
4215
4216
4217
4218
4219
4220
4221
4222
4223
4224
4225
4226
4227
4228
4229
4230
4231
4232
4233
4234
4235
4236
4237
4238
4239
4240
4241
4242
4243
4244
4245
4246
4247
4248
4249
4250
4251
4252
4253
4254
4255
4256
4257
4258
4259
4260
4261
4262
4263
4264
4265
4266
4267
4268
4269
4270
4271
4272
4273
4274
4275
4276
4277
4278
4279
4280
4281
4282
4283
4284
4285
4286
4287
4288
4289
4290
4291
4292
4293
4294
4295
4296
4297
4298
4299
4300
4301
4302
4303
4304
4305
4306
4307
4308
4309
4310
4311
4312
4313
4314
4315
4316
4317
4318
4319
4320
4321
4322
4323
4324
4325
4326
4327
4328
4329
4330
4331
4332
4333
4334
4335
4336
4337
4338
4339
4340
4341
4342
4343
4344
4345
4346
4347
4348
4349
4350
4351
4352
4353
4354
4355
4356
4357
4358
4359
4360
4361
4362
4363
4364
4365
4366
4367
4368
4369
4370
4371
4372
4373
4374
4375
4376
4377
4378
4379
4380
4381
4382
4383
4384
4385
4386
4387
4388
4389
4390
4391
4392
4393
4394
4395
4396
4397
4398
4399
4400
4401
4402
4403
4404
4405
4406
4407
4408
4409
4410
4411
4412
4413
4414
4415
4416
4417
4418
4419
4420
4421
4422
4423
4424
4425
4426
4427
4428
4429
4430
4431
4432
4433
4434
4435
4436
4437
4438
4439
4440
4441
4442
4443
4444
4445
4446
4447
4448
4449
4450
4451
4452
4453
4454
4455
4456
4457
4458
4459
4460
4461
4462
4463
4464
4465
4466
4467
4468
4469
4470
4471
4472
4473
4474
4475
4476
4477
4478
4479
4480
4481
4482
4483
4484
4485
4486
4487
4488
4489
4490
4491
4492
4493
4494
4495
4496
4497
4498
4499
4500
4501
4502
4503
4504
4505
4506
4507
4508
4509
4510
4511
4512
4513
4514
4515
4516
4517
4518
4519
4520
4521
4522
4523
4524
4525
4526
4527
4528
4529
4530
4531
4532
4533
4534
4535
4536
4537
4538
4539
4540
4541
4542
4543
4544
4545
4546
4547
4548
4549
4550
4551
4552
4553
4554
4555
4556
4557
4558
4559
4560
4561
4562
4563
4564
4565
4566
4567
4568
4569
4570
4571
4572
4573
4574
4575
4576
4577
4578
4579
4580
4581
4582
4583
4584
4585
4586
4587
4588
4589
4590
4591
4592
4593
4594
4595
4596
4597
4598
4599
4600
4