Towards Advanced Mathematical Reasoning for LLMs via First-Order Logic Theorem Proving

Chuxue Cao[1], **Mengze Li**[1], **Juntao Dai**[2], **Jinluan Yang**[3], **Zijian Zhao**[1]
Shengyu Zhang[3], **Weijie Shi**[1], **Chengzhong Liu**[1], **Sirui Han**[1*†], **Yike Guo**[1*]
[1]Hong Kong University of Science and Technology
[2]Peking University [3]Zhejiang University
ccaoai@connect.ust.hk {siruihan, yikeguo}@ust.hk

Abstract

Large language models (LLMs) have shown promising first-order logic (FOL) reasoning capabilities with applications in various areas. However, their effectiveness in complex mathematical reasoning involving multi-step FOL deductions is still under-researched. While LLMs perform competitively on established mathematical reasoning benchmarks, they struggle with multi-step FOL tasks, as demonstrated by Deepseek-Prover-V2-7B's low accuracy (4.2%) on our proposed theorem proving dataset. This issue arises from the limited exploration of diverse proof strategies and the potential for early reasoning mistakes to undermine entire proofs. To address these issues, we propose DREAM, a self-adaptive solution that enhances the Diversity and REAsonability of LLMs' generation strategies. DREAM incorporates an Axiom-Driven Strategy Diversification mechanism to promote varied strategic outcomes and a Sub-Proposition Error Feedback to help LLMs reflect on and correct their proofs. Our contributions include pioneering advancements in LLMs' mathematical reasoning through FOL theorem proving, introducing a novel inference stage solution that improves performance by 0.6% to 6.4%, and providing a curated dataset of 447 mathematical theorems in Lean 4 format for evaluation. Our code is available [1].

1 Introduction

Large language models (LLMs) have demonstrated emerging capabilities in first-order logic (FOL) reasoning (Zhou et al., 2024c), with successful applications across legal precedent analysis (Alam et al., 2023) and logical fallacy detection (Lalwani et al., 2025; Ibragimov et al., 2025a). However, their efficiency in addressing complex mathematical reasoning tasks characterized by multi-step FOL deductions remains underexplored (Cao et al., 2021).

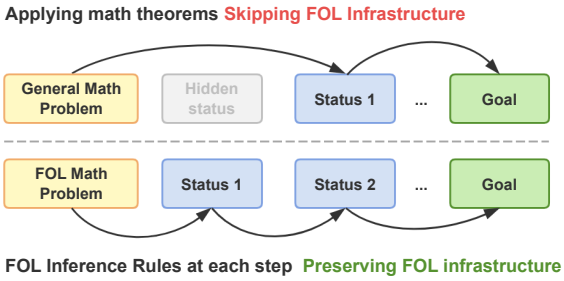

Figure 1: Distinction between First-Order Logic (FOL) and general math problems: FOL theorem proving requires strict stepwise adherence to FOL inference rules (e.g., universal instantiation, existential elimination), whereas general mathematical proving can utilize domain-specific mathematical theorems without explicitly referencing FOL infrastructure (including FOL rules and theorems).

While contemporary LLMs attain competitive performance on established formal mathematical reasoning benchmarks such as miniF2F (formal Olympiad-level mathematics (Zheng et al., 2022)) and ProofNet (formal undergraduate-level mathematics (Azerbayev et al., 2023)), they reveal persistent deficiencies in mathematical reasoning with multi-step FOL deductions. Our controlled experiments demonstrate that DeepSeek-Prover-V2-7B (Ren et al., 2025) — despite comprehensive pretraining on Lean 4's formal mathematics corpus— achieves merely 4.2% accuracy (pass@10) on our proposed FOL-based mathematical theorem proving tasks. This stark contrast between general and FOL-based mathematical reasoning capability, as shown in Figure 1, exposes limitations in current LLMs' capacity for handling nested quantifier interactions and negation propagation through extended deductive sequences (Qi et al., 2025).

For the FOL theorem proving problems, existing LLMs face two challenges: (i) *The Adaptive Strat-*

*Corresponding author. †Project leader.
[1]https://github.com/chuxuecao/dream-fol-prover

Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing, pages 12441–12461
November 4-9, 2025

egy Starvation Dilemma: Unlike standard mathematical problems, where fixed solution methods often suffice, FOL proofs demand both tactical flexibility and strategic oversight. The high sensitivity of proof chains to initial assumptions requires exploring multiple proof strategies and maintaining logical consistency throughout the deduction process. But current training paradigms predominantly utilize fixed logical structures from proof assistant libraries (e.g., Lean 4's Mathlib) (Lin et al., 2025), which encapsulate only a constrained subset of FOL applications, further preventing models from capturing the whole combinatorial space of potential logical constructions and reasoning patterns. While inference-stage solutions for general-domain math theorem proving are proposed to mitigate this issue (Yang et al., 2023; Zhao et al., 2024), they overlook the specific features of FOL proving, restricting their efficiency. (ii) *The Severe Cascading Error:* Within the reasoning chains for FOL theorem proving, early strategic errors can propagate through subsequent inferences, further undermining the entire proof, which can be defined as a cascading error (Kovács and Voronkov, 2013; Barwise, 1977). Compared with the modular error propagation seen in numerical calculations and code generation, the cascading error in FOL is more challenging due to the interdependence of logical steps and the lack of clear boundaries between errors. Thus, low-level error signals from a formal compiler are insufficient, as they fail to address the broader implications of flawed strategies.

To address the above challenges, we propose a novel inference stage solution that promotes the **D**iversity and **REA**sonability of LLM**s**' generation strategies, assisted by the detected errors across the entire proof, named **DREAM**. It includes two key designs: **Axiom-Driven Strategy Diversification:** To avoid strategy starvation, we propose an axiom-driven strategy diversification mechanism based on a k-wise combinational axiom tree. This approach enables diverse strategy selection by focusing on different axioms, resulting in varied strategic outcomes. **Sub-Proposition Error Feedback:** To mitigate cascading errors, we propose a sub-proposition error feedback mechanism that aligns each error message with its corresponding sub-proposition using inline comments. This approach provides insights into the sub-propositions, encouraging LLMs to reflect on and revise their proof strategies thoroughly.

Our contributions are summarized as three-fold:

- To the best of our knowledge, we are the first to advance LLMs' mathematical reasoning via FOL theorem proving, which especially requires LLMs to generate proof steps by strictly adhering to FOL rules and theorems.
- We propose an inference stage solution through axiom-driven strategy diversification and sub-proposition error feedback mechanisms to enhance LLM's performance in this challenging FOL theorem proving task, achieving average gains from 0.6% to 6.4%.
- A carefully curated dataset is provided for extensive evaluation, containing 447 mathematical theorems from 10 categories within first-order logic written in Lean 4 format.

2 Related Work

2.1 First-Order Logic Reasoning

The interaction between FOL reasoning and LLMs manifests in two key directions: (i) leveraging FOL to enhance the faithfulness of LLM reasoning and (ii) evaluating LLM's long-chain deduction capabilities. Recent advancements illustrate this dual focus. For instance, LOGIC-LM (Pan et al., 2023) and LINC (Olausson et al., 2023) employ LLMs to translate natural language (NL) statements to formal FOL expressions, then utilize symbolic reasoning tools for verification and self-refinement, thereby grounding LLM outputs in rigorous logical frameworks. Concurrently, studies such as Ryu et al. (2025), Qi et al. (2025), and Thatikonda et al. (2025) propose algorithms for constructing high-quality FOL datasets and evaluating LLMs' multi-step reasoning capabilities.

However, while these works mark significant progress, their datasets predominantly center on real-world scenarios (e.g., everyday life (Han et al., 2024; Tian et al., 2021; Saparov and He, 2023; Tafjord et al., 2021; Clark et al., 2020)). A critical gap persists in formal FOL mathematical reasoning. Despite efforts to evaluate LLMs' logical skills via NL-encoded FOL problems (Ibragimov et al., 2025b), their capacity to handle formal axiomatically defined systems (e.g., mathematical theorems, formal proof chains, or abstract logical relationships) remains underexplored. This omission limits understanding of LLMs' ability to navigate domains where precision, symbolic rigor, and adherence to axiomatic structures are paramount. To fill this gap, we create a formal FOL reasoning

dataset in the mathematical domain by utilizing the advanced FOL translation capabilities of LLMs. The detailed comparison between our datasets and previous datasets can be shown in Table 1.

2.2 Formal Theorem Proving

LLM-based theorem proving methods offer flexible control over problem complexity and diversity (Johansson and Smallbone, 2023; Zhou et al., 2024b; Wu et al., 2022; He et al., 2024; Wan et al., 2024; Xiong et al., 2023). Research in this area splits into two main approaches: complete proof generation and stepwise generation. For stepwise generation, models like BFS-Prover (Xin et al., 2025) and InternLM2.5-StepProver (Wu et al., 2024) predict proof steps based on current status, while LeanDojo reduces hallucination through retrieval-based premise selection (Yang et al., 2023). In contrast, LEGO-prover and DTV focus on prompting LLMs for complete proofs (Wang et al., 2023; Zhou et al., 2024a). Baldur enhances proof accuracy using error feedback (First et al., 2023), and Zhao et al. (2024) introduces a subgoal-based framework for LLMs. However, these methods have not optimized LLM's abilities in FOL reasoning by fully leveraging LLMs' specialized mathematical knowledge or utilizing the formal compiler effectively. Our work addresses this gap through axiom-driven strategy diversification and sub-proposition error feedback.

3 Preliminary & Motivation

3.1 Preliminary

We treat proof generation as a sequence-to-sequence task. Given a formal FOL theorem (x), which includes relevant axioms that describe the features of the concepts mentioned in the theorem, our goal is to generate a formal proof (y) that can be automatically verified by the formal compiler $compile(.)$ (De Moura et al., 2015). A proof is correct if it produces no error message, denoted as $compile(y) = pass$. Given a set of theorems $\{x_i\}_{i=1}^{N}$, the optimization goal for this task is to prove as many theorems as possible.

The objective function can be defined as Eq. 1:

$$max \sum_{i=1}^{N} \mathbb{I}(compile(y_i) = pass), \qquad (1)$$

where N is the total number of theorems attempted, y_i is the r-th proof for theorem x_i, and $\mathbb{I}$ is an indicator function that equals 1 if the proof is correct and 0 otherwise.

3.2 Motivation

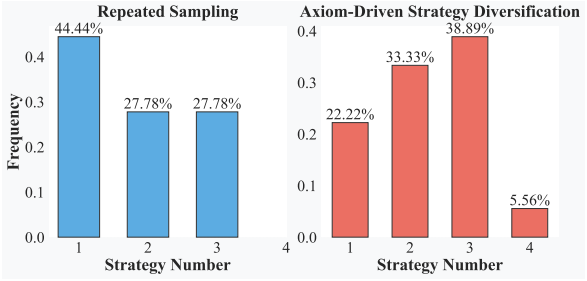

Figure 2: Comparison of strategy number distribution for six generated solutions tested on Claude 3.5.

Strategy Diversity: We first explore the effect of strategy diversity on LLM's ability for FOL theorem proving tasks. Figure 2 reveals that repeated sampling often yields repetitive proof strategies. Since FOL deduction relies on the stepwise application of logical rules and relevant axioms or lemmas, the lack of strategy diversity will severely restrict the search space for LLMs to discover valid solutions. To mitigate this homogeneity, we experimentally investigated whether explicitly guiding LLMs to prioritize distinct axiom combinations during proof generation could break this uniformity. Our experiments demonstrate that such targeted axiom-focused prompting significantly diversifies the generated strategies under fixed computational budgets (Figure 2), unlocking latent reasoning pathways. This finding motivates our proposed axiom-driven strategy diversification module, which systematically exploits axiom relevance to enhance exploration while maintaining logical coherence. Examples of diverse strategies generated by focusing on different sets of axioms are shown in Appendix I.

Cascading Error: Another key factor influencing the reasonableness of FOL proofs is the cascading error (see examples in Appendix J). That's because simply providing LLMs with error messages from the formal compiler yields minimal improvement since resolving such errors demands revisions to the entire proof. To address this issue, we explore the effectiveness of mapping errors to specific sub-propositions within the proof, as shown in Figure 3. From the results, we can observe that the sub-proposition error feedback demonstrated significant enhancement over direct error feedback, motivating our proposal for a sub-proposition error feedback module to target corrections and mitigate cascading failures by linking errors to their corresponding logical components.

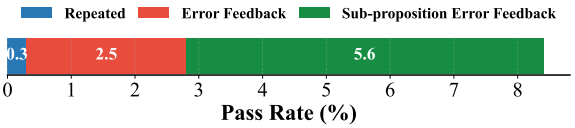

Figure 3: Pass rate on FOL theorem proving tasks using repeated sampling, error feedback, and sub-proposition error feedback. The longer rectangle is preferred.

4 Method

4.1 Overall Framework

In this section, we elaborate on the FOL theorem-proving framework that comprises two key components: axiom-driven strategy diversification and sub-proposition error feedback. **(i)** The axiom-driven strategy diversification aims to encourage the LLM to explore different ways of proving the theorem. To achieve this goal, given a theorem x, we can construct a k-wise combinatorial axiom tree to update strategies, which are executed through two or three times of revisions to prevent the LLM from getting stuck in the same incorrect reasoning; **(ii)** The sub-proposition error feedback aims to further ensure the reasonability of reasoning chains during theorem proving, which takes advantage of back-propagating error messages from previous failed proofs. We create sub-proposition error feedback that enhances self-correction by linking these error messages to sub-propositions. The model learns from sub-proposition errors of earlier attempts at each revision time. The overall framework is illustrated in Figure 4.

4.2 Axiom-Driven Strategy Diversification

To address the adaptive strategy starvation, we aim to expand the strategy search space by constructing a k-wise combinatorial axiom tree. Similar techniques can also be shown in Wang et al. (2024), which focuses on the LLM planning. This tree allows LLMs to systematically explore various strategies, improving their success rate.

Denote the LLM as θ and p_θ as the probability distribution from the LLM. We can initially generate a set of first-level axioms based on the context and the conjecture. The $O = \{o_1, o_2, \ldots, o_M\}$ are defined as axioms, where O is sampled from the distribution $p_\theta(\cdot \mid x)$, o_i denotes an individual axiom, and M is the number of axioms.

Specifically, we employ the k-wise combinatorial generation tree, where the second-level axioms O' are generated based on these first-level axioms. Each second-level axiom o'_s is a leaf node of the k-wise combinatorial generation tree and can be derived from one possible k-wise combination of the first-level axioms. We can denote the second-level axioms as Eq. 2, where S_k stands for the indexes of all possible k-wise combinations from M axioms in the first-level as Eq. 3. The number of elements in $\mathcal{S}_k$, $\binom{M}{k}$, represents the number of ways to choose k elements from a set of M distinct elements. Strategy $\mathcal{P}$ is generated using a new second-level axiom set o'_s. We can donate the strategy as Eq. 4.

$$\mathcal{O}' = \left\{ o'_s \,\middle|\, o'_s \sim p_\theta(\cdot \mid \{o_{s_i}\}_{i=1}^{k};\, x),\ s \in \mathcal{S}_k \right\} \tag{2}$$

$$\mathcal{S}_k = \left\{ s = (s_1, \ldots, s_k) \,\middle|\, 1 \le s_1 < \cdots < s_k \le M \right\} \tag{3}$$

$$\mathcal{P} \sim p_\theta(\,\cdot \,|x;\, o'_s) \tag{4}$$

4.3 Sub-proposition Error Feedback

To address the cascading error propagation inherent in formal proof correction, we leverage error feedback from formal verification compilers to iteratively refine LLM-generated proofs. However, FOL theorem proofs have numerous sub-propositions linked using logical connections like conjunction $\wedge$ and disjunction $\vee$. Directly applying word-level error messages generated by the formal compiler may not lead LLMs to create the linkage modifications between sub-propositions of first-order logic, seriously damaging the thorough proof correction. Thus, we propose the sub-proposition-level error feedback where the error messages are strictly aligned with the sub-propositions of the proof.

Denote the set of all previous $r-1$ failed attempts as $E = \{E_1, E_2, \ldots, E_{r-1}\}$. Each attempt contains a formal proof of the statement y_i and corresponding error messages $e_i = Compiler(y_i)$, where $Compiler(\cdot)$ is the formal compiler. We represent the aligned proofs y' using inline comments, placing sub-proposition annotations before the code block and error messages after the corresponding error line. y'_i is generated by an sub-proposition annotator L, $y'_i = L(E_i)$. An analyzer A examines mistakes at the sub-propositional level, offering insights for the r-th revision I_r into error patterns and suggesting strategies for improvement. We denote I_r as Eq. 5, where r stands for the current revision time.

$$I_r = A(\, x;\, \{y'_i\}_{i=1}^{r-1}\,) \tag{5}$$

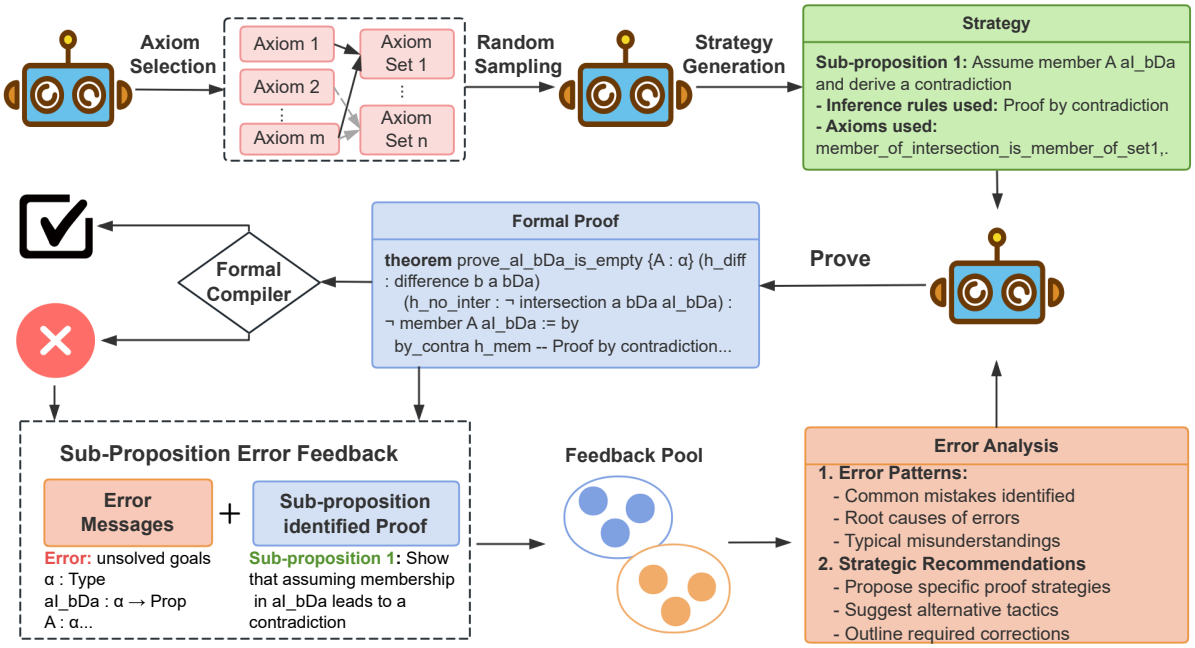

Figure 4: The overall pipeline of our proposed method. Given a conjecture, our method first applies axiom-driven strategy diversification to construct an axiom tree. Then, an axiom set is sampled from the second-level axiom tree for strategy generation. A proof is then generated based on this strategy. Incorrect proofs are labeled with sub-propositions and error messages from the formal compiler and placed into a feedback pool. Finally, an analysis of error patterns is conducted to provide strategic recommendations for iteratively refining the next round of proving.

The proof of current revision time y_r is generated as Eq. 6:

$$y_r \sim p_\theta(\cdot \,|x;\ I_r; \{E_j\}_{j=1}^{r-1}) \tag{6}$$

where p_θ represents the generative model, x denotes the theorem, I_r signifies the insight, and $\{E_j\}_{j=1}^{r-1}$ represents the collection of previous proofs with corresponding error messages generated by the compiler.

5 Experiment

5.1 Experimental Setup

Baselines: We adopt two well-known inference-stage solutions as comparisons to display the effectiveness of our method: (i) Repeated sampling (Repeated), where the LLM generates a correct proof for a theorem until it either reaches the maximum attempts or passes the formal compiler; (ii) Subgoal-based demonstration learning (Subgoal), which breaks down the theorem into subgoals in natural language and selects relevant examples for in-text demonstration learning (Zhao et al., 2024).

Evaluation Metric: Following Zhao et al. (2024), we select the cumulative pass rate as the metric for evaluation, which is the proportion of theorems solved at least once. A large pass rate is preferred.

Evaluation Dataset: Due to the lack of a mathematical evaluation benchmark with multi-step FOL deductions, we construct it as follows: **(i) TPTP Revised Dataset.** We converted 324 FOL problems from the TPTP format (Sutcliffe, 2017) to the Lean 4 format to support LLM proving. Specifically, we utilize LLMs to translate axioms and conjectures from TPTP to Lean 4 format, leveraging their exceptional expertise in Lean 4. Similar to prior autoformalization approaches (Zhang et al., 2024a; Yang et al., 2024), the dataset construction pipeline is illustrated in Figure 5. It involves three key steps: *Step 1: Lean 4 Format Translation:* We employ DeepSeek-V3 to convert conjectures and their associated axioms from TPTP to Lean 4 format. The translation prompt is detailed in Appendix 18, and each translated example is verified with the Lean 4 compiler (De Moura et al., 2015). *Step 2: Post-processing:* To facilitate LLM proving, we separate conjecture definitions from their context, add necessary import statements, and manually review the content for quality assurance. *Step 3: Context Optimization:* To improve LLM comprehension, we use DeepSeek-V3 to retain only essential contextual elements. The formal

compiler then verifies the simplified problems; **(ii) Manually Collected Dataset:** We curated a new dataset featuring 123 problems to cover various topics in the FOL theorem proving theme. Specifically, we manually collect theorems from academic papers and discrete mathematics textbooks. These theorems were converted to LaTeX and verified as valid first-order logic statements. They were then transformed from LaTeX to Lean 4 format using DeepSeek-V3, with up to 60 attempts. The dataset emphasizes intuitionistic logic, set theory, and computability, covering realizability, model theory, substitution, tautologies, and relationships between logical systems. Two human verifiers reviewed the annotations and corrected any inaccuracies.

Dataset	Creation	Domain	Formal	Division
RuleTaker (Clark et al., 2020)	Synthetic	Real-world	✗	✗
ProofWriter (Tafjord et al., 2021)	Synthetic	Real-world	✗	✗
LogicNLI (Tian et al., 2021)	Synthetic	Real-world	✗	✗
ProntonQA (Saparov and He, 2023)	Synthetic	Real-world	✓	✗
FOLIO (Han et al., 2024)	Manual	Real-world	✓	✗
ProverQA (Qi et al., 2025)	Synthetic	Real-world	✓	✗
Our Proposed Dataset	Synthetic & Manual	Mathematics	✓	✓

Table 1: Comparison between our mathematical FOL reasoning dataset and existing FOL datasets. "Formal" indicates the inclusion of a formal format, while "Division" refers to the subcategories within the dataset.

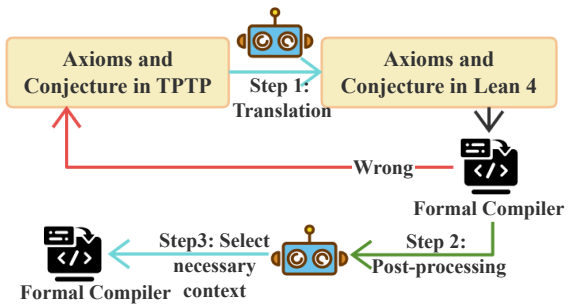

Figure 5: TPTP revision pipeline.

Implementation Details: We employ Claude 3.5 Sonnet (Anthropic, 2024) and DeepSeek-Prover-V2-7B (OpenAI, 2024) as the LLMs. For FOL theorem proving tasks, we utilize a 2-level, 2-wise combinatorial axiom tree, generating three to five axiom nodes at the first level. The maximum number of attempts is set to 10. Axiom-driven diversification is applied in the 4th and 7th revisions.

5.2 Performance Comparison

As shown in Table 2 and Table 3, we have the following key findings:

(i) Despite extensive training on formal mathematical proving materials, the LLMs tested in our dataset still performed poorly, highlighting the challenging nature of our proposed dataset. Claude 3.5 achieves a mere 0.2% pass rate, while DeepSeek-Prover-V2-7B reaches only 4.2%. The models perform relatively better on the manually collected dataset, which encompasses a broader range of mathematical topics and includes shorter contexts with fewer logical restrictions. This observation suggests that LLMs struggle with reasoning under strict logical constraints, such as FOL rules and axioms.

(ii) Our proposed DREAM significantly outperforms other methods on the FOL theorem proving task, achieving an average pass rate of 10.1% using Claude 3.5 and 8.3% using DeepSeek-Prover-V2-7B. Specifically, DREAM demonstrated its superior performance across all domains, showing its efficiency in FOL theorem proving. The repeated sampling method underperforms because of its limited search space on strategies, which prevents it from exploring more possible solutions, leading to repeated failures on the same errors. The subgoal-based demonstration learning method has introduced subgoal decompositions and demonstration examples. However, this approach overlooks FOL logic's non-modular error propagation characteristic, addressing errors only within specific modules. In comparison, our method uses a k-wise combinatorial axiom tree, allowing for systematic exploration of strategies. The specially designed axiom-driven strategy diversification has guaranteed its stable performance by systemically exploring different strategies. In contrast, the sub-proposition error feedback designed according to the feature of first-order logic stably guides the LLMs to the correct proof pathway. These mechanisms have resulted in the strong generalization ability of our method, making it well-suited for more diverse FOL theorem-proving tasks.

(iii) We also monitor the pass rates of various approaches, as shown in Figure 6, where we adopt different methods on our dataset using Claude 3.5. Initially, our method may not have performed as well as the subgoal-based demonstration learning method on the TPTP revised dataset. However, after the fourth revision, it began to show significant improvement. This trend suggests that our approach benefits from the iterative learning process, where each revision builds on the last. Our method achieves the top rank significantly after the second revision on the manually collected dataset, while the Subgoal-based demonstration learning

Models	Methods	FLD1	FLD2	GEO6	GEO8	GEO9	GRP5	NUM9	KRS1	SET1	Avg.
Claude 3.5	Repeated (Brown et al., 2024)	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	1.5%	0.0%	0.2%
	Subgoal (Zhao et al., 2024)	5.2%	3.1%	4.5%	**4.9%**	12.5%	0.0%	0.0%	3.0%	0.0%	3.7%
	DREAM(Ours)	**14.3%**	**12.5%**	**13.6%**	0.0%	0.0%	**20.0%**	**5.6%**	**3.0%**	**22.2%**	**10.1%**
DeepSeek-Prover-V2-7B	Repeated (Brown et al., 2024)	1.3%	0.0%	6.8%	7.3%	0.0%	10.0%	0.0%	**1.5%**	11.1%	4.2%
	Subgoal (Zhao et al., 2024)	0.0%	**3.1%**	9.1%	4.9%	**25.0%**	20.0%	**5.6%**	**1.5%**	0.0%	7.7%
	DREAM(Ours)	**3.9%**	0.0%	**11.4%**	**9.8%**	12.5%	**30.0%**	**5.6%**	**1.5%**	0.0%	**8.3%**

Table 2: Performance comparison on the TPTP revised dataset. "Avg." refers to the average pass rate (%).

Models	Methods	Avg.
Claude 3.5	Repeated (Brown et al., 2024)	32.5%
	Subgoal (Zhao et al., 2024)	27.6%
	DREAM(Ours)	**41.5%**
DeepSeek-Prover-V2-7B	Repeated (Brown et al., 2024)	12.2%
	Subgoal (Zhao et al., 2024)	**22.8%**
	DREAM(Ours)	21.1%

Table 3: Performance comparison on the manually collected dataset. "Avg." denotes the average pass rate (%).

method ranks the lowest. This result demonstrates the strong generalization ability of our method across different types of problems.

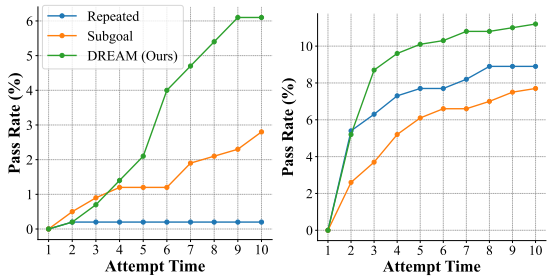

Figure 6: Passing rate comparisons on Claude 3.5 for various methods on the TPTP revision dataset (left) and the manually collected dataset (right) across attempts. The x-axis indicates the attempt number. Our proposed method achieves the highest passing rate starting from the fourth attempt on the TPTP revised dataset and the third attempt on the manually collected dataset.

5.3 Ablation Studies

To further understand the effectiveness of our proposed DREAM, we analyze the factors that influence its efficiency based on the TPTP revised dataset. Overall, as shown in Table 4, we can observe that DREAM achieves the highest performance across most domains. This performance underlines the effectiveness of various modules as follows.

Analysis on axiom-driven strategy diversification. The effectiveness of the axiom-driven strategy diversification is evident when we analyze its removal from our full method. Without it, the pass rate decreased from 10.1% to 9.9% for Claude 3.5 and from 8.3% to 3.2% for DeepSeek-Prover-V2-7B. Except for domains like GEO8, GEO9, and SET1, its absence generally leads to lower pass rates. However, using this mechanism alone does not guarantee improved performance, as Claude 3.5's pass rate increased, while DeepSeek-Prover-V2-7B's decreased.

Analysis on Sub-proposition Error Feedback. The lack of sub-proposition error feedback has resulted in a significant decrease in the average pass rate, dropping from 10.1% to 5.6%, with all domains showing a notable decline. This decline may be because the sub-propositions in the proofs allow LLMs to analyze the explored strategies, refine the decomposition of the main theorem's sub-propositions, and offer targeted revision insights.

5.4 Discussion about Background Restrictions

In our experiments, we also observed an interesting phenomenon related to the background restrictions for the axiom. Specifically, we can remove standard mathematical axioms (FLD, GEO, GRP, NUM, SET) to analyze LLMs' internal abilities for knowledge recall. Using 32 axiom-free problems (4 per domain from 8 TPTP domains), Claude 3.5's success rate increased while DeepSeek-Prover-V2-7B's declined compared to axiom-dependent scenarios (Table 5). This contrast suggests Claude owns broader mathematical knowledge as a general LLM, flexibly applying familiar axioms. However, the training of DeepSeek-Prover-V2-7B relies on complete proofs related to predefined backgrounds, leading to a model excelling in structured contexts but struggling when axioms are removed. The divergence highlights how training data (specialized proofs vs general knowledge) shapes FOL problem-solving approaches.

5.5 Case Studies

We provide two cases related to strategy diversity and sub-proposition error feedback to visualize the effectiveness of our method in solving FOL theorem proving problems. Figure 7 shows that our method successfully generates the correct proof after applying the axiom-driven strategy diversi-

Models	SD	SE	FLD1	FLD2	GEO6	GEO8	GEO9	GRP5	NUM9	KRS1	SET1	Avg.
Claude 3.5	-	-	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	1.5%	0.0%	0.2%
	✓	-	13.0%	0.0%	9.1%	2.4%	0.0%	0.0%	0.0%	3.0%	11.1%	5.6%
	-	✓	3.9%	3.1%	11.4%	2.4%	**12.5%**	**20.0%**	0.0%	3.0%	**33.3%**	9.9%
	✓	✓	**14.3%**	**12.5%**	**13.6%**	0.0%	0.0%	**20.0%**	**5.6%**	**3.0%**	22.2%	**10.1%**
DeepSeek-Prover-V2-7B	-	-	1.3%	0.0%	6.8%	7.3%	0.0%	10.0%	0.0%	**1.5%**	**11.1%**	4.2%
	✓	-	0.0%	0.0%	2.3%	4.9%	**12.5%**	10.0%	2.8%	1.5%	0.0%	3.8%
	-	✓	0.0%	0.0%	2.3%	2.4%	0.0%	**20.0%**	2.8%	1.5%	**0.0%**	3.2%
	✓	✓	**3.9%**	0.0%	**11.4%**	**9.8%**	**12.5%**	**30.0%**	**5.6%**	**1.5%**	0.0%	**8.3%**

Table 4: Ablation results on TPTP revised dataset. "SD" stands for axiom-driven strategy diversification. "SE" stands for sub-proposition error feedback. "Avg." refers to the average pass rate (%).

fication. The strategy is derived from the totality_of_order_relation axiom's key feature of providing a disjunctive conclusion. This strategy, "Apply totality_of_order_relation to a and a", is directly implemented in the solution through have h := totality_of_order_relation A less_or_equal defined a a h_def h_def, where both resulting cases yield the desired reflexivity property. Figure 8 illustrates how incorporating sub-proposition error feedback enables the LLMs to make high-level modifications, resulting in successful proofs.

Context:

```
axiom symmetry_of_equality (X Y : A) :
  equalish Y X → equalish X Y
axiom totality_of_order_relation (X Y : A) :
  defined X → defined Y → less_or_equal X Y ∨ less_or_equal Y X
...
```

Conjecture:

```
theorem reflexivity_of_order_relation : less_or_equal a a := sorry
```

Selected Axioms: totality_of_order_relation…

Strategy:

Sub-proposition 1: Apply totality_of_order_relation to a and a
- Inference rules used: Direct application
- Axioms used:
  - totality_of_order_relation
  - a_is_defined

Solution:

```
theorem reflexivity_of_order_relation
…
(h_def : defined a) : less_or_equal a a := by
-- Apply totality of order relation to get a ≤ a ∨ a ≤ a
have h := totality_of_order_relation A less_or_equal defined a a
h_def h_def
-- Both cases give us exactly what we want to prove
cases h with
| inl h_left => exact h_left -- First case: a ≤ a directly
| inr h_right => exact h_right -- Second case: a ≤ a directly
```

Figure 7: Case study: The effect of strategy diversification on proving a theorem for Claude 3.5.

Sub-proposition error feedback

```
theorem prove_reflexivity (a : G) : equalish a a := by
/- /tmp/tmpfxlrz700:1:36: error: function expected at equalish term
has type?m.9 -/

  --Sub-proposition 1: multiply identity a = a holds (establishing the
first equation needed for total_function2)
  have h1 : multiply identity a = a := left_identity a
/- /tmp/tmpfxlrz700:3:39: error: unknown identifier 'left_identity' -/
/- /tmp/tmpfxlrz700:3:12: error: unknown identifier 'multiply' -/

  --Sub-proposition 2: multiply identity a = a holds again (establishing
the second equation needed for total_function2)
  have h2 : multiply identity a = a := left_identity a
/- /tmp/tmpfxlrz700:5:39: error: unknown identifier 'left_identity' -/
/- /tmp/tmpfxlrz700:5:12: error: unknown identifier 'multiply' -/
…
```

Revised correct proof

```
theorem prove_reflexivity
…
-- First establish multiply identity a = a
  have h1 : multiply identity a = a := left_identity a
  -- Use the same equation again
  have h2 : multiply identity a = a := left_identity a
  -- Since both h1 and h2 show multiply identity a = a,
  -- we can use total_function2 to prove equalish a a
  exact total_function2 h1 h2
```

Figure 8: Case study: The effect of sub-proposition error feedback on proving a theorem for Claude 3.5.

Models	Methods	w/o Background	w Background
Claude 3.5	Repeated (Brown et al., 2024)	12.5%	0.0%
	Subgoal (Zhao et al., 2024)	18.8%	3.1%
	DREAM(Ours)	21.9%	12.5%
DeepSeek-Prover-V2-7B	Repeated (Brown et al., 2024)	0.0%	9.4%
	Subgoal (Zhao et al., 2024)	3.1%	6.3%
	DREAM(Ours)	3.1%	9.4%

Table 5: The exploration is related to the background constraint for axioms. We randomly selected 32 samples from the TPTP revised FOL-based dataset.

6 Conclusion

This work first advances LLMs' mathematical reasoning abilities via first-order theorem proving. Through detailed TPTP revision and manual collection, we curated a challenging FOL theorem-proving dataset that uncovers the drawbacks of existing formal LLMs (e.g., DeepSeek-Prover-V2-7B). Moreover, we propose the DREAM framework, a novel inference stage solution incorporating axiom-driven strategy diversification and sub-proposition error feedback for efficient FOL theorem proving. Our approach effectively addresses the limitations of LLMs in handling mathematical first-order logic-proving tasks in formal formats. Extensive experiments can verify the effectiveness of DREAM over previous methods for this challenging and complex reasoning task.

7 Limitations

Our experiments demonstrate the effectiveness of DREAM in enhancing LLMs' performance in FOL-based theorem-proving tasks, more diverse FOL-based mathematical tasks could be considered in the future. Additionally, the experimental results show a consistent increase in performance, even by the 10th revision. However, due to resource limitations, we have no chance to extend the experiment to identify our method's saturation point. Future research should also account for the model's internal structured reasoning patterns (Wen et al., 2025). In addition to performance, the ethical and societal acceptability, such as safety, honesty and value, should also be incorporated to enhance the controllability and reliability of reasoning (Cao et al., 2025; Yang et al., 2025; Ju et al., 2025).

Acknowledgments

This work is funded in part by the HKUST Start-up Fund (R9911), Theme-based Research Scheme grant (T45-205/21-N), the InnoHK initiative of the Innovation and Technology Commission of the Hong Kong Special Administrative Region Government, and the research funding under HKUST-DXM AI for Finance Joint Laboratory (DXM25EG01).

References

Mohammad Nazmul Alam, Md. Shahin Kabir, and Arun Verma. 2023. Data and knowledge engineering for legal precedents using first-order predicate logic. In *2023 4th IEEE Global Conference for Advancement in Technology (GCAT)*, pages 1–8.

AI Anthropic. 2024. Claude 3.5 sonnet model card addendum. *Claude-3.5 Model Card*.

Zhangir Azerbayev, Bartosz Piotrowski, Hailey Schoelkopf, Edward W. Ayers, Dragomir Radev, and Jeremy Avigad. 2023. Proofnet: Autoformalizing and formally proving undergraduate-level mathematics. *Preprint*, arXiv:2302.12433.

Jon Barwise. 1977. An introduction to first-order logic. In *Studies in Logic and the Foundations of Mathematics*, volume 90, pages 5–46. Elsevier.

Bradley Brown, Jordan Juravsky, Ryan Ehrlich, Ronald Clark, Quoc V. Le, Christopher Ré, and Azalia Mirhoseini. 2024. Large language monkeys: Scaling inference compute with repeated sampling. *Preprint*, arXiv:2407.21787.

Chuxue Cao, Han Zhu, Jiaming Ji, Qichao Sun, Zhenghao Zhu, Yinyu Wu, Juntao Dai, Yaodong Yang, Sirui Han, and Yike Guo. 2025. Safelawbench: Towards safe alignment of large language models. *arXiv preprint arXiv:2506.06636*.

Feng Cao, Yang Xu, Jun Liu, Shuwei Chen, and Jianbing Yi. 2021. A multi-clause dynamic deduction algorithm based on standard contradiction separation rule. *Information Sciences*, 566:281–299.

Peter Clark, Oyvind Tafjord, and Kyle Richardson. 2020. Transformers as soft reasoners over language. *arXiv preprint arXiv:2002.05867*.

Leonardo De Moura, Soonho Kong, Jeremy Avigad, Floris Van Doorn, and Jakob von Raumer. 2015. The lean theorem prover (system description). In *Automated Deduction-CADE-25: 25th International Conference on Automated Deduction, Berlin, Germany, August 1-7, 2015, Proceedings 25*, pages 378–388. Springer.

Emily First, Markus N. Rabe, Talia Ringer, and Yuriy Brun. 2023. Baldur: Whole-proof generation and repair with large language models. In *Proceedings of the 31st ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering*, ESEC/FSE 2023, page 1229–1241, New York, NY, USA. Association for Computing Machinery.

Simeng Han, Hailey Schoelkopf, Yilun Zhao, Zhenting Qi, Martin Riddell, Wenfei Zhou, James Coady, David Peng, Yujie Qiao, Luke Benson, Lucy Sun, Alex Wardle-Solano, Hannah Szabo, Ekaterina Zubova, Matthew Burtell, Jonathan Fan, Yixin Liu, Brian Wong, Malcolm Sailor, Ansong Ni, Linyong Nan, Jungo Kasai, Tao Yu, Rui Zhang, Alexander R. Fabbri, Wojciech Kryscinski, Semih Yavuz, Ye Liu, Xi Victoria Lin, Shafiq Joty, Yingbo Zhou, Caiming Xiong, Rex Ying, Arman Cohan, and Dragomir Radev. 2024. Folio: Natural language reasoning with first-order logic. *Preprint*, arXiv:2209.00840.

Yuhang He, Jihai Zhang, Jianzhu Bao, Fangquan Lin, Cheng Yang, Bing Qin, Ruifeng Xu, and Wotao Yin. 2024. Bc-prover: Backward chaining prover for formal theorem proving. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 3059–3077.

Shokhrukh Ibragimov, Arnulf Jentzen, and Benno Kuckuck. 2025a. On the logical skills of large language models: evaluations using arbitrarily complex first-order logic problems. *arXiv preprint arXiv:2502.14180*.

Shokhrukh Ibragimov, Arnulf Jentzen, and Benno Kuckuck. 2025b. On the logical skills of large language models: evaluations using arbitrarily complex first-order logic problems. *Preprint*, arXiv:2502.14180.

Moa Johansson and Nicholas Smallbone. 2023. Exploring mathematical conjecturing with large language models. In *NeSy*, pages 62–77.

Chengyi Ju, Weijie Shi, Chengzhong Liu, Jiaming Ji, Jipeng Zhang, Ruiyuan Zhang, Jiajie Xu, Yaodong Yang, Sirui Han, and Yike Guo. 2025. Benchmarking multi-national value alignment for large language models. In *Findings of the Association for Computational Linguistics: ACL 2025*, pages 20042–20058, Vienna, Austria. Association for Computational Linguistics.

Laura Kovács and Andrei Voronkov. 2013. First-order theorem proving and vampire. In *International Conference on Computer Aided Verification*, pages 1–35. Springer.

Abhinav Lalwani, Tasha Kim, Lovish Chopra, Christopher Hahn, Zhijing Jin, and Mrinmaya Sachan. 2025. Autoformalizing natural language to first-order logic: A case study in logical fallacy detection. *Preprint*, arXiv:2405.02318.

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

Jianqiao Lu, Yingjia Wan, Zhengying Liu, Yinya Huang, Jing Xiong, Chengwu Liu, Jianhao Shen, Hui Jin, Jipeng Zhang, Haiming Wang, et al. 2024. Process-driven autoformalization in lean 4. *arXiv preprint arXiv:2406.01940*.

Theo Olausson, Alex Gu, Ben Lipkin, Cedegao Zhang, Armando Solar-Lezama, Joshua Tenenbaum, and Roger Levy. 2023. Linc: A neurosymbolic approach for logical reasoning by combining language models with first-order logic provers. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 5153–5176.

OpenAI. 2024. Openai o1 system card.

Liangming Pan, Alon Albalak, Xinyi Wang, and William Yang Wang. 2023. Logic-lm: Empowering large language models with symbolic solvers for faithful logical reasoning. *Preprint*, arXiv:2305.12295.

Chengwen Qi, Ren Ma, Bowen Li, He Du, Binyuan Hui, Jinwang Wu, Yuanjun Laili, and Conghui He. 2025. Large language models meet symbolic provers for logical reasoning evaluation. In *ICLR*. ICLR.

Z. Z. Ren, Zhihong Shao, Junxiao Song, Huajian Xin, Haocheng Wang, Wanjia Zhao, Liyue Zhang, Zhe Fu, Qihao Zhu, Dejian Yang, Z. F. Wu, Zhibin Gou, Shirong Ma, Hongxuan Tang, Yuxuan Liu, Wenjun Gao, Daya Guo, and Chong Ruan. 2025. Deepseek-prover-v2: Advancing formal mathematical reasoning via reinforcement learning for subgoal decomposition. *Preprint*, arXiv:2504.21801.

Hyun Ryu, Gyeongman Kim, Hyemin S. Lee, and Eunho Yang. 2025. Divide and translate: Compositional first-order logic translation and verification for complex logical reasoning. In *The Thirteenth International Conference on Learning Representations*.

Abulhair Saparov and He He. 2023. Language models are greedy reasoners: A systematic formal analysis of chain-of-thought. In *The Eleventh International Conference on Learning Representations*.

Geoff Sutcliffe. 2017. The tptp problem library and associated infrastructure. *Journal of Automated Reasoning*, 59(4):483–502.

Oyvind Tafjord, Bhavana Dalvi, and Peter Clark. 2021. ProofWriter: Generating implications, proofs, and abductive statements over natural language. In *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*, pages 3621–3634, Online. Association for Computational Linguistics.

Ramya Keerthy Thatikonda, Wray Buntine, and Ehsan Shareghi. 2025. Assessing the alignment of fol closeness metrics with human judgement. *Preprint*, arXiv:2501.08613.

Jidong Tian, Yitian Li, Wenqing Chen, Liqiang Xiao, Hao He, and Yaohui Jin. 2021. Diagnosing the first-order logical reasoning ability through LogicNLI. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 3738–3747, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Yuxuan Wan, Wenxuan Wang, Yiliu Yang, Youliang Yuan, Jen-tse Huang, Pinjia He, Wenxiang Jiao, and Michael Lyu. 2024. Logicasker: Evaluating and improving the logical reasoning ability of large language models. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 2124–2155.

Evan Z Wang, Federico Cassano, Catherine Wu, Yunfeng Bai, William Song, Vaskar Nath, Ziwen Han, Sean M. Hendryx, Summer Yue, and Hugh Zhang. 2024. Planning in natural language improves LLM search for code generation. In *The First Workshop on System-2 Reasoning at Scale, NeurIPS'24*.

Haiming Wang, Huajian Xin, Chuanyang Zheng, Lin Li, Zhengying Liu, Qingxing Cao, Yinya Huang, Jing Xiong, Han Shi, Enze Xie, et al. 2023. Lego-prover: Neural theorem proving with growing libraries. *arXiv preprint arXiv:2310.00656*.

Pengcheng Wen, Jiaming Ji, Chi-Min Chan, Juntao Dai, Donghai Hong, Yaodong Yang, Sirui Han, and Yike Guo. 2025. Thinkpatterns-21k: A systematic study on the impact of thinking patterns in llms. *arXiv preprint arXiv:2503.12918*.

Yuhuai Wu, Albert Q. Jiang, Wenda Li, Markus N. Rabe, Charles Staats, Mateja Jamnik, and Christian Szegedy. 2022. Autoformalization with large language models. *Preprint*, arXiv:2205.12615.

Zijian Wu, Suozhi Huang, Zhejian Zhou, Huaiyuan Ying, Jiayu Wang, Dahua Lin, and Kai Chen. 2024. Internlm2.5-stepprover: Advancing automated theorem proving via expert iteration on large-scale lean problems. *Preprint*, arXiv:2410.15700.

Ran Xin, Chenguang Xi, Jie Yang, Feng Chen, Hang Wu, Xia Xiao, Yifan Sun, Shen Zheng, and Kai Shen. 2025. Bfs-prover: Scalable best-first tree search for llm-based automatic theorem proving. *Preprint*, arXiv:2502.03438.

Jing Xiong, Jianhao Shen, Ye Yuan, Haiming Wang, Yichun Yin, Zhengying Liu, Lin Li, Zhijiang Guo, Qingxing Cao, Yinya Huang, et al. 2023. Trigo: Benchmarking formal mathematical proof reduction for generative language models. In *2023 Conference on Empirical Methods in Natural Language Processing (EMNLP 2023)*, pages 11594–11632. Association for Computational Linguistics (ACL).

Jinluan Yang, Dingnan Jin, Anke Tang, Li Shen, Didi Zhu, Zhengyu Chen, Ziyu Zhao, Daixin Wang, Qing Cui, Zhiqiang Zhang, et al. 2025. Mix data or merge models? balancing the helpfulness, honesty, and harmlessness of large language model via model merging. *arXiv preprint arXiv:2502.06876*.

Kaiyu Yang, Aidan Swope, Alex Gu, Rahul Chalamala, Peiyang Song, Shixing Yu, Saad Godil, Ryan Prenger, and Anima Anandkumar. 2023. LeanDojo: Theorem proving with retrieval-augmented language models. In *Neural Information Processing Systems (NeurIPS)*.

Yuan Yang, Siheng Xiong, Ali Payani, Ehsan Shareghi, and Faramarz Fekri. 2024. Harnessing the power of large language models for natural language to first-order logic translation. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 6942–6959, Bangkok, Thailand. Association for Computational Linguistics.

Lan Zhang, Xin Quan, and Andre Freitas. 2024a. Consistent autoformalization for constructing mathematical libraries. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 4020–4033, Miami, Florida, USA. Association for Computational Linguistics.

Lan Zhang, Xin Quan, and Andre Freitas. 2024b. Consistent autoformalization for constructing mathematical libraries. *arXiv preprint arXiv:2410.04194*.

Xueliang Zhao, Wenda Li, and Lingpeng Kong. 2024. Subgoal-based demonstration learning for formal theorem proving. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pages 60832–60865. PMLR.

Kunhao Zheng, Jesse Michael Han, and Stanislas Polu. 2022. minif2f: a cross-system benchmark for formal olympiad-level mathematics. In *International Conference on Learning Representations*.

Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric Xing, et al. 2023. Judging llm-as-a-judge with mt-bench and chatbot arena. *Advances in neural information processing systems*, 36:46595–46623.

Jin Peng Zhou, Charles Staats, Wenda Li, Christian Szegedy, Kilian Q Weinberger, and Yuhuai Wu. 2024a. Don't trust: Verify–grounding llm quantitative reasoning with autoformalization. *arXiv preprint arXiv:2403.18120*.

Jin Peng Zhou, Yuhuai Wu, Qiyang Li, and Roger Grosse. 2024b. Refactor: Learning to extract theorems from proofs. *Preprint*, arXiv:2402.17032.

Ruiwen Zhou, Wenyue Hua, Liangming Pan, Sitao Cheng, Xiaobao Wu, En Yu, and William Yang Wang. 2024c. Rulearena: A benchmark for rule-guided reasoning with llms in real-world scenarios. *arXiv preprint arXiv:2412.08972*.

A Pseudocode

The pseudocode for our proposed method is presented in Table 6.

Algorithm: ProveTheorem

Input: Conjecture x, LLM θ, max revision R
Output: Proof y

$O \leftarrow$ GenerateFirstLevelAxioms(x, θ)
$O' \leftarrow$ GenerateSecondLevelAxioms(O, x, θ, k)
$E \leftarrow \{\}$ // Initialize error collection
for $r \leftarrow 1$ **to** R **do**
  $o'_s \leftarrow$ SelectAxioms(O')
  if $r = 0$ **then**
    $y_r \leftarrow$ GenerateInitialProof(x, θ)
  if $r = 4$ or $r = 7$ **then**
    $o' \leftarrow$ SelectSecondLevelAxioms(O')
    $P \leftarrow$ GenerateStrategy(x, o')
    $y_r \leftarrow$ GenerateProofBasedOnStrategy(x, P, θ)
  else
    $I_r \leftarrow$ AnalyzeWithFeedback(x, o'_s, $\{y'_i\}_{i=1}^{r-1}$)
    $y_r \leftarrow$ GenerateRevisedProof(x, I_r, E, θ)
  if compile(y_r) = pass **then**
    return y_r
  $e_r \leftarrow$ compile(y_r)
  $y'_r \leftarrow$ AnnotateProof(y_r, e_r)
  $E \leftarrow E \cup \{(y_r, e_r)\}$
return NULL

Table 6: Core pseudocode of DREAM for FOL theorem proving.

B Dataset Statistics

The domain distribution from the TPTP library is shown in Table 7

Domain	Description	TPTP	Lean4
FLD1	Field Theory	136	77
FLD2	Field Theory	143	32
GEO6	Geometry	97	44
GEO8	Geometry	57	41
GEO9	Geometry	43	8
GRP5	Group Theory	10	10
NUM9	Number Theory	36	36
KRS1	Knowledge Representation	94	67
SET1	Set Theory	11	9
Total		627	324

Table 7: First-order theorems extracted from the TPTP library (Sutcliffe, 2017).

C Data Quality Control

Similar to prior autoformalization approaches like DeepSeek-Prover-V2 (Ren et al., 2025; Zhang et al., 2024b; Lu et al., 2024), we employ Lean verification to ensure logical accuracy and mitigate potential biases in LLM-assisted dataset creation.

To assess the reliability of the manual verification stage, we conducted a human annotation study measuring inter-annotator agreement. Two experts in formal reasoning independently evaluated 40 problems—20 from the manually collected corpus and 20 from TPTP-Revised—for correctness and logical consistency of theorem statements and their Lean 4 formalizations. As shown in Table **??**, the overall problem-wise agreement was 82.5%. This is consistent with the 81% human agreement reported by Zheng et al. (2023), supporting the reliability of our dataset.

D Detailed Statistics of Dataset Construction

We provide detailed statistics for both steps of the dataset construction pipeline in Table 9.

E Computational Efficiency

This section provides a detailed analysis of the computational budget and runtime overhead. We evaluate performance under two evaluation paradigms: (1) fixed maximum iteration count, and (2) fixed LLM call budget.

E.1 Comparison Under Fixed Iteration Budget

To ensure a fair comparison with prior work such as Subgoal, which also employs iterative refinement within a bounded number of steps, we set the maximum number of iterations to 10 for all methods. Table **??** reports the average number of LLM calls and token consumption using Claude 3.5, along with proof success rates across benchmark categories. Despite requiring more LLM calls per iteration due to internal branching and feedback mechanisms, DREAM achieves significantly higher overall accuracy (12.7% vs. 6.7%).

E.2 Comparison Under Fixed LLM Call Budget

To further assess efficiency, we compare DREAM and Subgoal under a constrained total budget of 17 LLM calls—the average number used by Subgoal in the 10-iteration setup. This ensures equivalence in resource usage while isolating algorithmic effectiveness. As shown in Table **??**, DREAM maintains superior performance even under reduced budget, achieving an average accuracy of 11.6%, com-

Annotators	Correct Rate (Manual)	Correct Rate (TPTP-Revised)	Avg. Correct Rate
Annotator 1	90%	85%	87.5%
Annotator 2	90%	95%	92.5%
Problem-wise Agreement	85%	80%	82.5%

Table 8: Problem-wise agreement between two annotators on manually collected and TPTP-revised problems.

Category	Original	Success Translations	Success Translation Rate (%)	Avg. Success Translation Time	Success Optimizations	Success Optimization Rate (%)
FLD001	136	96	70.6	4.0	77	80.2
FLD002	143	38	26.6	5.0	32	84.2
GEO006	97	57	58.8	8.0	44	77.2
GEO008	57	45	78.9	2.0	41	91.1
GEO009	43	13	30.2	5.0	8	61.5
GRP005	10	10	100.0	2.0	10	100.0
NUM009	36	36	100.0	1.0	36	100.0
KRS001	94	75	79.8	2.0	67	89.3
SET001	11	9	81.8	5.0	9	100.0
Average	627	379	67.4	4.0	324	85.5

Table 9: Details of TPTP-Revised dataset construction. Translation refers to converting the TPTP theorem to Lean 4, while Optimization involves selecting the necessary context.

Method	Avg. Calls	Avg. Tokens	FLD1	FLD2	GEO6	GEO8	GEO9	GRP5	NUM9	KRS1	SET1	Manual	Avg.
Subgoal	18	53,378	5.2%	3.1%	4.5%	4.9%	10.0%	0.0%	7.1%	3.0%	0.0%	29.1%	6.7%
DREAM	26	73,184	14.3%	12.5%	13.6%	0.0%	0.0%	20.0%	0.0%	3.0%	22.2%	41.5%	12.7%

Table 10: Computational budget comparison over 10 iterations using Claude 3.5. "Avg. Calls" denotes the average number of LLM calls per solution, and "Avg. Token" represents the average token consumption per solution.

pared to 6.5% for Subgoal. This demonstrates that DREAM's structured reasoning framework yields higher utility per call.

F Quantitative Error Analysis

To gain deeper insight into the sources of failure in formal proof generation, we conducted a fine-grained categorization of errors present in unsuccessful proofs produced by DREAM. We employed DeepSeek-V3 as an auxiliary classifier to automatically identify and label error types based on the Lean 4 rejection messages and surrounding proof context. Only error categories occurring more than once are included in the analysis to ensure meaningful interpretation.

The results, summarized in Table **??**, reveal that the majority of errors stem from syntactic and type-level inconsistencies. This highlights a key limitation of large language models in generating precise formal expressions. Future improvements could therefore benefit from syntax-aware decoding strategies or post-hoc correction modules designed to enforce consistency in types and identifiers.

G Dataset Examples

An example from our dataset is shown in Table 9.

H Illustrative Reasoning Tree and Axiom Expansion

We provide a simplified visualization of the hierarchical axiom expansion process used in DREAM. Starting from the target theorem, first-level axioms are retrieved based on semantic relevance. Then, second-level axioms are generated by combining $k = 2$ first-level axioms, each leading to a distinct proof strategy.

I Examples of Axiom-Driven Strategy Diversification

Figures 11 to 15 illustrate how focusing on different axioms results in varied proving strategies.

J Cascading Error in FOL Proofs

Illustrative examples of cascading errors in FOL proofs are presented in Figures 16 and 17.

K TPTP to Lean 4 Format Conversion Prompt

The prompt for converting TPTP to Lean 4 format is shown in Table 18.

Method	Max Calls	FLD1	FLD2	GEO6	GEO8	GEO9	GRP5	NUM9	KRS1	SET1	Manual	Avg.
Subgoal	17	3.9%	3.1%	4.5%	4.9%	10.0%	0.0%	7.1%	3.0%	0.0%	28.2%	6.5%
DREAM	17	10.4%	9.4%	13.6%	0.0%	0.0%	20.0%	0.0%	3.0%	22.2%	37.4%	11.6%

Table 11: Accuracy comparison under equal LLM call budget (17 calls) using Claude 3.5.

Error Type	Frequency
type mismatch	151
unknown identifier	92
incorrect application	26
incorrect tactic	25
unknown tactic	10
unsolved goals	7
unsolved metavariable	6
invalid syntax	6
incomplete proof	3
proof strategy error	3
declaration conflict	2
invalid constructor	2
missing dependency	2
missing instance	2

Table 12: Distribution of error types in proofs generated by DREAM using Claude 3.5. Only error types with frequency greater than 1 are displayed.

FOL-MATH Problem Example

```
import Mathlib
import Aesop
variable (α : Type)
variable (member : α → α → Prop)
variable (subset : α → α → Prop)
variable (equal_sets : α → α → Prop)
variable (intersection : α → α → α → Prop)
variable (h : α → α → α → α)
variable (a b c alb blc albIc : α)
axiom member_of_intersection_is_member_of_set1 (Element Set1 Set2 Intersection : α) :
¬ intersection Set1 Set2 Intersection ∨ ¬ member Element Intersection ∨ member Element Set1
axiom member_of_intersection_is_member_of_set2 (Element Set1 Set2 Intersection : α) :
¬ intersection Set1 Set2 Intersection ∨ ¬ member Element Intersection ∨ member Element Set2
axiom member_of_both_is_member_of_intersection (Element Set1 Set2 Intersection : α) :
¬ intersection Set1 Set2 Intersection ∨ ¬ member Element Set1 ∨ ¬ member Element Set2 ∨ member Element Intersection
axiom intersection_axiom1 (Set1 Set2 Intersection : α) :
member (h Set1 Set2 Intersection) Intersection ∨
intersection Set1 Set2 Intersection ∨
member (h Set1 Set2 Intersection) Set1
axiom intersection_axiom2 (Set1 Set2 Intersection : α) :
member (h Set1 Set2 Intersection) Intersection ∨
intersection Set1 Set2 Intersection ∨
member (h Set1 Set2 Intersection) Set2
axiom intersection_axiom3 (Set1 Set2 Intersection : α) :
¬ member (h Set1 Set2 Intersection) Intersection ∨
¬ member (h Set1 Set2 Intersection) Set2 ∨
¬ member (h Set1 Set2 Intersection) Set1 ∨
intersection Set1 Set2 Intersection
axiom a_intersection_b : intersection a b alb
axiom b_intersection_c : intersection b c blc
axiom a_intersection_blc : intersection a blc albIc

theorem prove_alb_intersection_c_is_albIc : ¬ intersection alb c albIc := sorry
```

Figure 9: An example of FOL-MATH dataset (SET006).

Visualization of a k-wise axiom tree

Root (Theorem)
|
|-- First-Level Axioms: {o_1, o_2, o_2, o_4, o_5}
|
|-- Second-Level Axioms (k=2 combinations):
  |-- o_{12} = combine(o_1, o_2) → Strategy P_1
  |-- o_{13} = combine(o_1, o_3) → Strategy P_2
  |-- o_{23} = combine(o_2, o_3) → Strategy P_3
  |-- ...

Theorem: theorem conjecture (A : Point) (L M N : Line) :
(apart_point_and_line A L ∧ ¬apart_point_and_line A M ∧ ¬apart_point_and_line A N ∧ ¬convergent_lines M L ∧ ¬convergent_lines N L) →¬distinct_lines M N := sorry

o_1: distinct_not_convergent (new): ∀ (X Y : Line), distinct_lines X Y → ¬convergent_lines X Y
o_2: not_convergent_distinct (new): ∀ (X Y : Line), ¬convergent_lines X Y → distinct_lines X Y
o_2: parallel_not_convergent (new): ∀ (X Y : Line), convergent_lines X Y → ¬distinct_lines X Y
o_4: not_convergent_parallel (new): ∀ (X Y : Line), ¬convergent_lines X Y → distinct_lines X Y
o_5: point_not_incident_parallel (new): ∀ (X : Point) (Y Z : Line),\napart_point_and_line X Y → convergent_lines Y Z → ¬convergent_lines X Z

Figure 10: Visualization of a k-wise axiom tree.

Context+Conjecture

```
import Mathlib
import Aesop
variable (Point : Type) (Line : Type)
variable (distinct_points : Point → Point → Prop)
variable (distinct_lines : Line → Line → Prop)
variable (convergent_lines : Line → Line → Prop)
variable (apart_point_and_line : Point → Line → Prop)
variable (intersection_point : Line → Line → Point)
axiom apart3 (X : Line) : ¬ convergent_lines X X
axiom apart6 {X Y Z : Line} :
convergent_lines X Y → (convergent_lines X Z ∨ convergent_lines Y Z)
axiom con2 {X Y : Line} {Z : Point} :
convergent_lines X Y →
(apart_point_and_line Z X ∨ apart_point_and_line Z Y) →
distinct_points Z (intersection_point X Y)
axiom ceq3 {X Y : Line} :
convergent_lines X Y → distinct_lines X Y

theorem conjecture {X Y Z : Line} :
convergent_lines X Y →
apart_point_and_line (intersection_point X Y) Z →
distinct_lines X Z ∧ distinct_lines Y Z := sorry
```

Figure 11: Context and conjecture of the demonstrative example, from which the LLMs generate the following two axiom sets.

Selected Axiom Set 1

(new) intersection_existence: For any lines X and Y, if they are convergent then their intersection point exists and is unique
Rationale: We need to ensure the intersection_point function is well-defined for convergent lines.
(new) point_on_line_distinct: For any point P and lines L1 L2, if P lies on L1 but not on L2, then L1 and L2 are distinct lines
Rationale: This helps establish that if a point (like the intersection point) lies on one line but is apart from another, those lines must be distinct.
These axioms together provide the necessary logical steps to prove that if an intersection point of two convergent lines is apart from a third line, then both original lines must be distinct from the third line.

Figure 12: The first set of axioms generated by the LLM based on the given context and conjecture.

Strategy based on the Selected Axiom Set 1

To prove that when two lines X and Y are convergent, and their intersection point is apart from a third line Z, both X and Z are distinct lines AND Y and Z are distinct lines, I'll break this into sub-propositions:
Sub-proposition 1: The intersection point of X and Y exists and is well-defined
- Inference rules: Direct application
- Axioms used: intersection_existence
- Justification: Since X and Y are convergent (given in hypothesis), their intersection point exists
Sub-proposition 2: X and Z are distinct lines
- Inference rules: Indirect proof using point_on_line_distinct
- Axioms used: point_on_line_distinct
- Justification: The intersection point lies on X (by intersection_existence) but is apart from Z (given in hypothesis), therefore X and Z must be distinct lines
Sub-proposition 3: Y and Z are distinct lines
- Inference rules: Indirect proof using point_on_line_distinct
- Axioms used: point_on_line_distinct
- Justification: The intersection point lies on Y (by intersection_existence) but is apart from Z (given in hypothesis), therefore Y and Z must be distinct lines
Main Strategy:
1. First establish that the intersection point exists and lies on both X and Y using intersection_existence
2. Use the hypothesis that this intersection point is apart from Z
3. Apply point_on_line_distinct twice:
   - Once for X and Z using the fact that the intersection point lies on X but is apart from Z
   - Once for Y and Z using the fact that the intersection point lies on Y but is apart from Z
4. Combine these results to form the conjunction required in the conclusion
This strategy relies heavily on the relationship between points lying on or being apart from lines and the distinctness of lines, which is exactly what point_on_line_distinct provides.

Figure 13: Strategy generated by prompting the LLM to focus on the first set of axioms, utilizing a direct proof method.

Selected Axiom set 2

(existing) ceq3: For any lines X and Y, if they are convergent then they are distinct
Rationale: This helps establish one part of the logical foundation - convergent lines must be distinct.
(new) intersection_existence: For any lines X and Y, if they are convergent then their intersection point exists and is unique
Rationale: We need to ensure the intersection_point function is well-defined for convergent lines.

Figure 14: The second set of axioms generated by the LLM based on the given context and conjecture.

Strategy based on the Selected Axiom Set 2

To prove that if lines X and Y are convergent and their intersection point is apart from line Z, then both X and Z are distinct lines, and Y and Z are distinct lines, I propose the following strategy:
1. First sub-proposition: distinct_lines X Y
- Inference rules: Direct application
- Axioms used: ceq3
- This establishes that X and Y are distinct lines when they are convergent
2. Second sub-proposition: convergent_lines X Y → (convergent_lines X Z ∨ convergent_lines Y Z)
- Inference rules: Direct application
- Axioms used: apart6
- This shows that Z must be convergent with either X or Y (or both)
3. Third sub-proposition: ¬convergent_lines X Z
- Inference rules: Proof by contradiction
- Axioms used: con2, apart3
- If X and Z were convergent, then their intersection point would exist
- This would contradict the given hypothesis that intersection_point X Y is apart from Z
4. Fourth sub-proposition: ¬convergent_lines Y Z
- Inference rules: Similar to sub-proposition 3
- Axioms used: con2, apart3
- Similar contradiction argument as above
5. Final sub-proposition: distinct_lines X Z ∧ distinct_lines Y Z
- Inference rules: Combining previous results
- Previously proven lemmas: Results from sub-propositions 3 and 4
- Axioms used: ceq3
- The non-convergence of X with Z and Y with Z implies distinctness through ceq3's contrapositive
The strategy relies heavily on showing that Z cannot be convergent with either X or Y, which then implies distinctness through the contrapositive of ceq3. The key insight is that if Z were convergent with either line, it would contradict the given hypothesis about the intersection point being apart from Z.

Figure 15: Strategy generated by prompting the LLM to focus on the first set of axioms, employing a proof by contradiction.

Cascading Error Example 1

```
import Mathlib
import Aesop
variable {A : Type}
...
variable (defined : A → Prop)
axiom existence_of_inverse_multiplication (X : A) :
  defined X → (¬equalish X additive_identity) →
  equalish (multiply X (multiplicative_inverse X)) multiplicative_identity
axiom commutativity_multiplication (X Y : A) :
  defined X → defined Y → equalish (multiply X Y) (multiply Y X)
...
axiom multiply_equals_k_11 : equalish (multiply a d) k
theorem multiply_not_equal_to_k_12 : ¬equalish (multiply b c) k := sorry
theorem multiply_not_equal_to_k_12 : ¬equalish (multiply b c) k := by
  -- Sub-proposition 1: Assume for contradiction that multiply b c equals k
  have h : ¬equalish (multiply b c) k := by
    intro h_contra
    -- Sub-proposition 2: Use the existence of multiplicative inverses to derive a contradiction
    have h1 := existence_of_inverse_multiplication multiplicative_identity 0
/- /tmp/tmpx4ncf3v6:41:51: error: unknown identifier 'multiplicative_identity' -/
    have h2 := existence_of_inverse_multiplication additive_identity 0
/- /tmp/tmpx4ncf3v6:42:51: error: unknown identifier 'additive_identity' -/
    have h3 := commutativity_multiplication multiplicative_identity multiplicative_identity
/- /tmp/tmpx4ncf3v6:43:68: error: unknown identifier 'multiplicative_identity' -/
/- /tmp/tmpx4ncf3v6:43:44: error: unknown identifier 'multiplicative_identity'-/
    have h4 := commutativity_multiplication additive_identity multiplicative_identity
    have h5 := commutativity_multiplication multiplicative_identity additive_identity
    have h6 := commutativity_multiplication additive_identity additive_identity
    have h7 := transitivity_of_equality multiplicative_identity multiplicative_identity additive_identity
    have h8 := transitivity_of_equality additive_identity multiplicative_identity multiplicative_identity
    have h9 := compatibility_of_equality_and_multiplication multiplicative_identity multiplicative_identity multiplicative_identity
    have h10 := compatibility_of_equality_and_multiplication additive_identity multiplicative_identity multiplicative_identity
    have h11 := compatibility_of_equality_and_multiplication multiplicative_identity additive_identity multiplicative_identity
    have h12 := compatibility_of_equality_and_multiplication additive_identity additive_identity multiplicative_identity
    have h13 := different_identities
    have h14 := a_is_defined
    have h15 := b_is_defined
    have h16 := c_is_defined
    have h17 := d_is_defined
    have h18 := k_is_defined
    have h19 := s_is_defined
    have h20 := b_not_equal_to_additive_identity
    have h21 := d_not_equal_to_additive_identity
    have h22 := multiply_equals_s_9
    have h23 := multiply_equals_s_10
    have h24 := multiply_equals_k_11
    -- Simplify all the assumptions and derive a contradiction
    simp_all
  -- Conclude the proof
  exact h
```

Figure 16: An illustrative example of cascading error.

Cascading Error Example 2

```
import Mathlib
import Aesop
variable {F : Type}
variable (add : F → F → F)
variable (multiply : F → F → F)
variable (additive_identity : F)
variable (multiplicative_identity : F)
variable (equalish : F → F → Prop)
variable (defined : F → Prop)
axiom existence_of_identity_multiplication (X : F) :
defined X → equalish (multiply multiplicative_identity X) X
axiom well_definedness_of_multiplication (X Y : F) :
defined X → defined Y → defined (multiply X Y)
axiom compatibility_of_equality_and_multiplication (X Y Z : F) :
defined Z → equalish X Y → equalish (multiply X Z) (multiply Y Z)
variable (a b d : F)
axiom a_is_defined : defined a
axiom b_is_defined : defined b
axiom d_is_defined : defined d
axiom a_equals_d : equalish a d
theorem multiply_not_equal_to_multiply :
¬equalish (multiply d b) (multiply a b) := by
/- /tmp/tmpatssqx03:23:43: error: unsolved goals
F : Type
multiply : F → F → F
equalish : F → F → Prop
a b d : F
⊢ ¬equalish (multiply d b) (multiply a b) -/
  --Sub-proposition 1: Multiplication by Identity Preserves Equality
  have h₁ := existence_of_identity_multiplication multiplicative_identity
/- /tmp/tmpatssqx03:25:50: error: unknown identifier 'multiplicative_identity'-/
  --Sub-proposition 2: Multiplication is Well-Defined
  have h₂ := well_definedness_of_multiplication multiplicative_identity a
  have h₃ := well_definedness_of_multiplication multiplicative_identity b
  --Sub-proposition 3: Multiplication is Compatible with Equality
  have h₄ := compatibility_of_equality_and_multiplication a multiplicative_identity b
  have h₅ := compatibility_of_equality_and_multiplication d multiplicative_identity b
  --Sub-proposition 4: `a` is Defined
  have h₆ := a_equals_d
  have h₇ := a_is_defined
  have h₈ := b_is_defined
  have h₉ := d_is_defined
  intro h
  simp_all
  <;> aesop
```

Figure 17: An illustrative example of cascading error.

```
System prompt:
Your task is to convert TPTP format axioms and conjectures into Lean
4 format. Follow these guidelines:

1. Type Declarations:
   - Declare all necessary types using `Type`
   - Define type variables when needed using uppercase letters (e.g.,
    `A`, `B`)

2. Axiom Conversion:
   - Convert each TPTP axiom into a complete Lean 4 definition
   - Use appropriate Lean 4 syntax for logical operators:
   - Do not use `sorry` in axiom definitions

3. Conjecture Conversion:
   - Convert the conjecture into a theorem statement
   - Use `theorem` for the declaration
   - End the theorem with `sorry`
   - Do not provide the proof

4. Code Format:
   - Wrap all Lean 4 code with ```lean``` markers
   - Use proper indentation
   - Include necessary imports
   - Add brief comments explaining complex translations

5. Variable Handling:
   - Declare all variables with appropriate types
   - Maintain consistent variable naming between axioms and
   conjecture
   - Use meaningful variable names when possible

Please ensure each conversion preserves the original logical meaning
while following Lean 4's syntax and type system.

User prompt:
Input TPTP Format:

Axioms:
{axioms}

Conjecture:
{conjecture}

Please provide the Lean 4 conversion following the guidelines above.
```

Figure 18: Prompts for converting first-order axioms and conjectures from TPTP format to Lean4 format.