Learning to Reason with Insight for Informal Theorem Proving

Anonymous ACL submission

Abstract

Although most of the automated theorem-proving approaches depend on formal proof systems, informal theorem proving can align better with large language models' (LLMs) strength in natural language processing. In this work, we identify a main bottleneck for informal theorem proving via LLMs as a lack of insight, namely the difficulty of recognizing the core techniques required to solve complex problems. To address this, we propose a novel framework designed to cultivate this essential reasoning skill and enable insightful reasoning in LLMs. We propose **DeepInsightTheorem**, a hierarchical dataset that structures informal proofs by explicitly extracting core techniques and proof sketches alongside the final proof. To fully exploit this dataset, we design a Progressive Multi-Stage SFT strategy that mimics the human learning process, guiding the model from basic proof writing to insightful thinking. Our experiments on challenging mathematical benchmarks demonstrate that this insight-aware generation strategy significantly outperforms baselines. These results demonstrate that teaching models to identify and apply core techniques can substantially improve their mathematical reasoning.

1 Introduction

Automated theorem proving (ATP) has long been a central goal in the field of artificial intelligence, serving as a key benchmark for evaluating machine reasoning. Recent progress in deep learning, particularly with Large Language Models (LLMs), has greatly changed the field of ATP. Most previous research attempts to solve this problem by combining LLMs with formal proof engines like Lean, Coq, and Isabelle (Zheng et al., 2022, Liu et al., 2023a, Tsoukalas et al., 2024) or by using specialized languages (Welleck et al., 2022). In contrast, informal theorem proving aims to generate proofs using natural language and standard mathematical notation, such as LaTeX. This approach aligns well with the strengths of modern LLMs.

However, only a few works have studied informal theorem proving, such as Welleck et al. (2022) and Zhang et al. (2025b), and the area remains highly underexplored. Most existing research has focused on framework construction and has rarely attended to the proof generation mechanism or the key bottlenecks of LLM-based informal theorem proving. Additional related work will be discussed in Appendix A.

In this paper, we emphasize the importance of core techniques in informal mathematical proofs. We identify in Section 2.2 and 3.1 that the bottlenecks of proof generation lie in the realization of the core techniques. Then in Section ref3, we further argue in principle that to improve a model's math reasoning, it is feasible and beneficial to review the discovery of the core technique from the original proof (acquisition) and cultivate insight by applying these learned techniques (application) during inference. To achieve this, we introduce DeepInsightTheorem, a hierarchical dataset in Section 4.1 based on the current framework DeepTheorem (Zhang et al., 2025b), and a corresponding progressive multi-stage training strategy that mirrors the human learning journey in Section 4.2, both for training LLM for insightful reasoning.

We evaluated the generated informal proof based on DeepTheorem and have a new design for insight evaluation. Through extensive experiments, we demonstrate that this insight-aware method significantly outperforms standard baselines in Section 6. The results confirm that by explicitly teaching the model to extract deep information, we can move beyond simple text imitation toward authentic, insight-driven mathematical reasoning.

In summary, the primary contributions of our work are as follows:

- **Identification of importance of the core tech-**

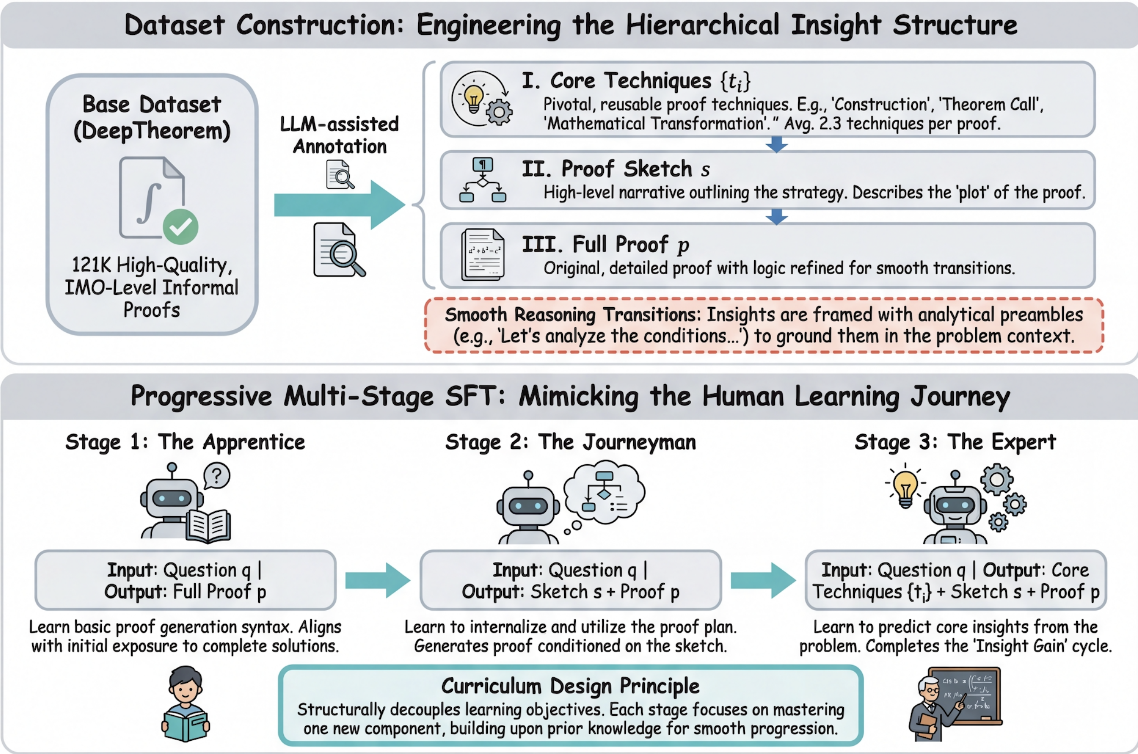

Figure 1: The realization of the dual process: the top figure shows the construction of DeepInsightTheorem, and the bottom performs the process of our multi-stage SFT.

niques: We show that the real perplexity for informal theorem proving is caused by the core technique realization.

- **The Introduction and Formalization of "Insight"**: We introduce the concept of mathematical insight and propose an insightful reasoning pattern. We show in principle that this approach improves mathematical reasoning upon appropriate training.
- **A Novel Hierarchical Insight Dataset**: We construct a hierarchical data structure by explicitly extracting core techniques from base datasets to assist with knowledge review and insightful reasoning.
- **A Progressive Training Curriculum for Cultivating Insight**: We design a progressive, multi-stage training regimen to accomplish a real insightful reasoning.

2 Preliminary

We consider the context of informal mathematical proof generation. Let $\mathcal{V}$ be the vocabulary, which is a finite discrete set. Then the sequence space $\mathcal{S}$ is defined as the union of the product spaces of any finite power $\mathcal{S} \triangleq \bigcup_{i=l}^{\infty} \mathcal{V}^l$.

We can then denote the space of theorems as a subspace $\mathcal{Q}$ of $\mathcal{S}$ which contains all well-defined math problems for theorem-proving. For any $q \in \mathcal{Q}$, we denote the space of proofs with respect to q as $\mathcal{P}(q) \subset \mathcal{S}$ that contains all valid proofs of q.

The Large language model (LLM) M_θ is an auto-regressive network which will predict the next token's distribution over the vocabulary when it is working. To train π_θ to be a theorem-prover, the training datasets are always collected from many resources of math corpus and commonly formed as theorem-proof pairs (q, p). For Supervised fine-tuning (SFT) on such dataset, the object is then to

$$\max_{\theta} \quad \mathbb{E}_{q \sim \mathcal{D}, p \sim \mathcal{D}_{P(q)}} \pi_\theta(p|q),$$

where $\mathcal{D}$ is the distribution of the math problems in training set and $\mathcal{D}_{P(q)}$ is the distribution over the valid proofs of each question in the training set.

2.1 Notion of Insight

Insight is not the proof itself, but the preliminary, high-level perception and foresight of the core techniques required to construct it. Insight is the cognitive act of identifying the pivotal idea (e.g., " apply the Pigeonhole Principle " , " utilize a specific invariant xxx ") through the given conditions to figure out the essence of solution. We call such ideas as core techniques in this paper. Generally, for theorem-proving, we have three main classes to group these techniques, which cover the most common and necessary techniques in math proofs:

- **Construction**: Introducing auxiliary objects. (e.g. " Define the sequence $x_{n+1} = \mathrm{T}x_n$ for $n \geq 0$. This Picard iteration constructs the sequence $x_0, x_1 = \mathrm{T}x_0, x_2 = \mathrm{T}^2x_0, \ldots$ ")
- **Theorem Call**: Invoking a known lemma, theorem or any existing results. (e.g. " By using Contor's Theorem ")
- **Mathematical Transformation**: Performing a smart mathematical manipulation to change the question to a new stage. (e.g. " Define a topology τ on the set of integers $\mathbb{Z}$... shifting the language and tools entirely from number theory to topology. ")

Formally, we denote the space of techniques as a subspace $\mathcal{T} \subset \mathcal{S}$ that contains all possible techniques in those three classes. Generally, we can then write each proof p of q as an ordered sequence

$$p = (r_1, t_1, r_2, t_2, \cdots, r_k), \quad (1)$$

where $t_i \in \mathcal{T}$ and r_j is a basic mathematical reasoning sentence.

2.2 Insightfulness Evaluation

Especially for expert mathematicians, when confronting a novel problem, they may quickly have a proof picture depending on their powerful insight cultivated from their accumulation. As for general LLM, instead it always fails to have such a high-level idea at a very initial stage of proof generation.

We evaluated several powerful commercial LLMs with nothink mode [1], including Gemini2.5-Flash (Gemini Team, 2025) and Deepseek-R1 (DeepSeek-AI, 2025), by prompting them to give insights for several math competition problems randomly selected from Putnam and FIMO datasets. Then we use o3-mini to review and evaluate the insights generated by these models from the point of depth and completeness. See Appendix E.1 for details. For both the insight-generation and evaluation prompts, see Appendix D for details.

It turns out that, on the depth side, both models basically failed to produce deep identifications, instead giving shallow guesses or vaguely hinting at some directions, and from the completeness side, they always only see one or two low-hanging fruit, missing more important techniques.

3 Methodology

3.1 Weak Recognition of Core Techniques

The main body of a proof is composed of easy-to-learn basic mathematical reasoning sentences. For a general LLM π_{TP} generating proof for a problem, the uncertainty keeps low until a core technique needs recognizing. The reasons are two-fold. One is that core techniques are intrinsically harder to recognize. The other is that the learning of LLM may focus on basic reasoning details due to its large percentage in a proof.

As we can see the generated proof of the well-known Baire Category Theorem in General Topology by Qwen2.5-14B-Instruct (Qwen Team, 2025) from Figure 2, tokens with a soaring entropy are always indicating core techniques:

Construction: "`Constructing a Sequence of ...`" by constructing a sequence of nested open sets

Theorem call: "`Using completeness...`" by using the completeness of the ground space.

Mathematical transformation: " `Consider the closures ...` " by transferring the issue of open sets to their closures.

The above three techniques respectively belong to those three main classes we set generally for math proofs as defined in Section 2.1. The rest of the proof are those basic mathematical reasoning combining these core techniques, making the whole proof complete and smooth.

When it needs to determine which technique to use, the π_{TP} has

$$\pi_{\text{TP}}(t_i | s_{i-1}, q) < \delta, \quad (2)$$

for some $\delta > 0$ and any $t_i \in \mathcal{T}(q)$, where s_{i-1} is the prefix before t_i. Hence it is hard for π_{TP} to see through the core techniques $t \in \mathcal{T}(q)$ at the very beginning:

$$\pi_{\text{TP}}(t|q) = \sum_s \pi_{\text{TP}}(t|s, q)\pi_{\text{TP}}(s|q) < \delta, \quad (3)$$

where s is over the possible prefix. Similarly, by recursively using (2), for the probability $\eta_{\text{TP}}(p, q)$ of π_{TP} generating a valid proof p (1) of q, we have $\eta_{\text{TP}}(p, q) = \mathcal{O}(\delta^k)$, which means the ability to complete a solid proof is restricted by the uncertainty. A detailed justification for this result can

[1] Long chain of thought would always make the model first go over the whole proof and then summarize the core techniques, which is against the thinking pattern of insight. So it is more appropriate to disable the thinking.

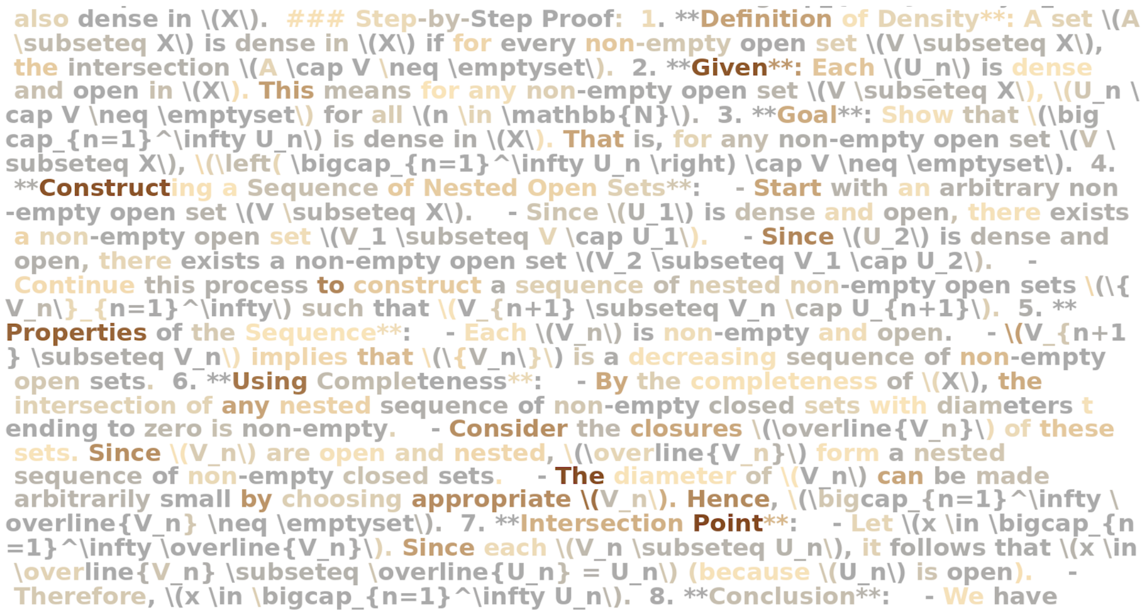

Figure 2: The generated proof of the Baire Category Theorem generated by Qwen2.5-14B-Instruct. From light grey to deep warm color, the token entropy increases. Most tokens are of low entropy, composing basic mathematical reasoning. Tokens with soaring entropy indicate core techniques.

be found in Appendix C. We call $\pi_{\text{TP}}(t|q)$ as the **insightfulness** of π_{TP} with respect to $t \in \mathcal{T}(q)$.

Therefore, successful realization of core techniques determines a valid proof generation for theorem-proving. Core techniques are more appropriate to be memory anchor for pattern recognition in complex mathematical reasoning. This prompts a necessity of enhanced learning for bridging question and its core techniques.

3.2 The Dual Process of Insight Acquisition and Application

It is natural to make a direct connection between the question and its resolving techniques if we want the LLM to see through the essence. After technique identification, it remains to integrate those core techniques, which then basically relies on the fundamental mathematical reasoning capability.

Actually this relates to the change of thinking pattern. For a training LLM π_θ , by the total probability decomposition, we have

$$\pi_\theta(t|q) = \sum_p \pi_\theta(t|p,q)\pi_\theta(p|q). \tag{4}$$

Different from (3), here we consider a whole proof generation p. The $\pi_\theta(p|q)$ term is exactly what π_θ learns from training on plain theorem-proof pairs, while the $\pi_\theta(t|p,q)$ term represents how well can π_θ summarize the core techniques from learning different proofs p of q and extract the common technique t. We then call $\pi_\theta(t|p,q)$ as the **acquisition** of π_θ on $t \in \mathcal{T}(q)$ from proof p.

There involve two complementary cognitive processes of human's after-class review for coursework. One is the core technique summary, and the other is the application of summarized information. The acquisition mirrors the summary. To apply the learned techniques, we modify the generation pattern of π_θ, making it first foresee the core techniques and then start generating proof. Then the generation of π_θ for proof (1) becomes

$$\pi_\theta(p|q) = \pi_\theta(t|q)\prod_i \pi_\theta(r_i|r_{\leq i-1}, t, q), \tag{5}$$

where $t = \{t_1, \cdots, t_k\}$ and $r^{\leq i-1} = \{r_1, \cdots r_{i-1}\}$.

The terms in the product can be well learned from usual mathematical corpus as we discussed in Section 3.1. Therefore, the performance of π_θ basically depends on how the first term $\pi_\theta(t|q)$ to be learned. We can show that if the term is well learned, the probability $\eta_\theta(p,q)$ of π_θ generating a valid proof p (1) of q is $\eta_\theta = \Omega((1-\epsilon-\delta_1)^k)$ for some small $\epsilon > 0$ and $\delta_1 > 0$. One can refer to the Appendix C for a detailed proof.

We call these cognitive processes the **dual processes** of insight acquisition and application. Hence π_θ will benefit from such dual process if trained appropriately. Next we will show how to demonstrate the processes in practice through two experimental operations respectively.

4 How to Train an Insightful LLM

4.1 Dataset Construction

Our hierarchical dataset is engineered to explicitly instantiate the review process, making the core techniques explicitly learnable. To ensure high quality and sufficient challenge, we construct our data based on the DeepTheorem (Zhang et al., 2025b) dataset, a recently introduced, large-scale resource for informal mathematical theorem proving.

Base Dataset. The DeepTheorem dataset provides a robust foundation consisting of 121K high-quality, informal theorem-proof pairs at IMO-level difficulty. Its construction involves a rigorous pipeline including sourcing from diverse corpora and implementing strict decontamination against major benchmarks (e.g., MATH (Hendrycks et al., 2021), AIME (AIME Dataset), miniF2F (Zheng et al., 2022)). Since DeepTheorem already contains many common datasets used for theorem-proving training, we believe the experiments based on it can efficiently and effectively establish a generalized average performance result for our methods. One can refer to Zhang et al. (2025b) for more details about the dataset.

DeepInsightTheorem. Since LLM itself struggles to extract the core techniques (Section 3.1), we then consider assisting via a rich data structure by explicitly feeding the LLM with the core techniques in training. Building upon DeepTheorem's theorem-proof pairs (q, p), we perform additional annotations to decompose each proof into a hierarchical structure, of which the skeleton looks as

$$\big(q, \underbrace{(t_1, \ldots, t_m)}_{\text{core techs}}, \underbrace{s}_{\text{sketch}}, \underbrace{p}_{\text{proof}}\big).$$

This process involves a meticulous, LLM-assisted analysis to extract the deep information in proof p, following a dedicated prompt to design each component as follows:

- **Core Techniques** t_i: Instead of only listing the core techniques, we first include a guiding statement as "`Let's analyze the conditions...`", then those core techniques are introduced with heuristic language, for example: "`The condition ... tells us to construct an auxiliary function...` " or "`... suggests using xxx Theorem.`". The core techniques are themselves finally summarized at the end of this component according to those three main classes.
- **Proof Sketch** s: We design an additional component between the core techniques and proof. This is an outlining proof strategy based on the core techniques. This part mainly serves as a connection of different levels of thoughts, making a smooth and coherent thinking process.
- **Proof** p: The original, detailed proof from the base dataset serves as the ground-truth instantiation.

We call our newly constructed dataset **DeepInsight-Theorem** to indicate the insight-induced hierarchical design derived from DeepTheorem. See Figure 1 top for an illustration and a particular example in Figure 9.

Smooth Reasoning Transitions. A critical design consideration in the construction of DeepInsight-Theorem is to avoid presenting the list of Core Techniques in isolation. We frame the insight generation within a short, analytical preamble, which acts as a localized micro Chain-of-Thought (Wei et al., 2022) focusing on condition analysis from the question. Throughout reasoning, the LLM may hence leads a more accurate insight prediction compared to directly outputting a list of techniques.

Review-Application Logical Transformation: There exists a subtle but crucial logical transformation between the review and application dual processes, even though the main contents throughout these two processes are basically the same. The former is backward-looking: " `Analyzing this proof, we see that xxx is the core tech.`", while the latter is forward-looking: " `Given these problem conditions, xxx might be a core tech.`" These two processes by our design are integrated into the LLM-assisted data annotation. Therefore we need to carefully tune the prompt for the assistant LLM to ensure the smoothness of the transformation. The specific prompts used are detailed in Appendix C.

Key Statistics. In our curated dataset of approximately 100K problems, each proof relies on 3.6 core techniques on average, with complex problems often employing 4 or more in combination as illustrated in Figure 3. A distribution of core techniques in each class is shown in Figure 5.

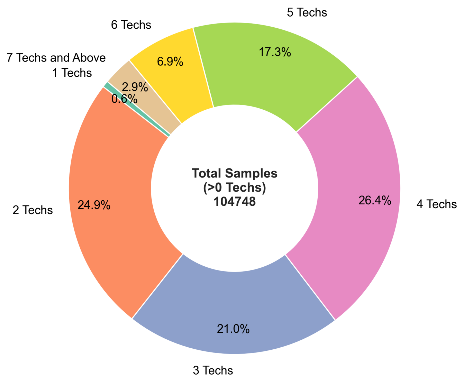

Figure 3: Distribution of number of core techniques

4.2 Progressive Multi-Stage SFT

The other process consists in the application of the core techniques summarized ahead (Section 3.2). The LLM can already have a hierarchical output due to the corresponding structure of DeepInsightTheorem and generate the core technique analysis first (Section 4.1). It remains to train LLM appropriately such that the two processes in (5) both work well. In principle, the decomposition (4) and (5) have already implied a two-stage training paradigm, i.e., the acquisition requires LLM to train on the original theorem-proof pairs (q, p) and the application to continue training on insight induced data.

We empirically show that two stages are necessary by comparing training solely on DeepInsightTheorem with the baseline. The results of Qwen2.5-7B and Llama3-8B are in Table 1.

It shows that LLM being directly trained on DeepInsightTheorem may not be able to gain benefits from the insight structure. One of the reasons is that higher value of (4) and (5) both assume a training on theorem-proof data to enlarge $\pi_\theta(p|q)$ and $\pi_\theta(r_i|r^{\leq i-1}, t, q)$ respectively. Missing the theorem-proof pair training it may not be able to have a good basic mathematical reasoning capability. On the other hand, this can also be viewed as a misalignment with the natural progression of learning: It may create a comprehension gap to directly teach a novice learner the expert's complex thinking pattern without first solidifying foundational proof-writing skills.

Hence we adopt a multi-stage training strategy to fine-tune LLMs. The following shows the main stages of the progressive SFT, each on the same but of different parts of the DeepInsightTheorem:

Methods	FIMO	Putnam	HMMT
Qwen2.5-7B	15.73	37.01	12.59
Llama3-8B	12.50	36.69	9.98

Table 1: Evaluation results for models solely trained on DeepInsightTheorem

Stage 1: The Apprentice (q, p) : Learn the basic task of proof generation on the plain theorem-proof pairs. This aligns with initial exposure to complete solutions, reflecting a junior learner referring the answer proof.

Stage 2: The Journeyman (q, s, p) : Learn to generate the proof plan on the theorem-sketch-proof triples. The model must now generate the proof conditioned on the high-level sketch, which begins the abstraction process. It reflects a senior learner's behavior where he can not only copy the proof but know the logic structure by steps.

Stage 3: The Expert $(q, \{t_i\}, s, p)$: Learn to insightfully predict core techniques from the problem and use them to derive the sketch and proof on the whole hierarchical data. This stage completes the dual processes, teaching the model to form the map from question directly towards the core techniques. It reflects hence the expert's thinking paradigm.

An illustration can be seen on Figure 1 bottom. Correspondingly, the progressive multi-stage design structurally decouples the learning objectives, providing a clear and focused target for each training stage. In each stage, the primary parameter updates are concentrated on learning the mapping between the problem and the newly introduced component. Since this mapping is not yet adequately learned by the model, continued training would not cause overfitting.

5 Evaluation

Since our task involves generating informal mathematical proofs by natural language, we adopt an LLM-as-Judge approach for evaluation. Following established practices in DeepTheorem, our evaluation is conducted on a set of challenging benchmarks to test the model's reasoning capability. Specifically, we use theorem-proving questions drawn from **FIMO** (Liu et al., 2023a), **PutnamBench** (Tsoukalas et al., 2024), and a newly constructed theorem-proving subset of the Harvard-MIT Mathematics Tournament (**HMMT**) (Tournament, 2025).

To assess the quality of the generated proofs, we follow the evaluation framework in DeepTheorem but omit a separate "correctness" judgment, which makes no sense for our task. Our evaluation centers on the following three core dimensions of the proof text itself (Zhang et al., 2025b):

- **Logical Validity**: Checking if each step follows logically from the previous one and indicating any logical errors.
- **Completeness**: Verifying if all necessary steps are included to fully prove the theorem.
- **Clarity**: Assessing if the proof is clear, unambiguous, and well-explained.

We use DeepSeek-R1 (DeepSeek-AI, 2025) as the judge model. For each generated proof, the judge is prompted to analyze and score it on a continuous scale from 0 to 1 for each of the three dimensions above. The final score for a proof is calculated as a weighted average of these three-dimensional scores. The specific weighting scheme and the detailed prompts used for this LLM-as-Judge evaluation are provided in Appendix D.

Since the model trained by our method would generate a formatted output containing three components, for the sake of fairness, we only extract the final proof component for evaluation against the base methods.

6 Experiments

We conduct supervised fine-tuning on two open-source model families: Qwen2.5 (Qwen Team, 2025) and Llama3 (Llama Team, 2024). For each family, we start from their respective base models (e.g., Qwen2.5-Base, Llama3.2-Base).

Baseline. For each base model, we establish a baseline by performing standard SFT on the question-proof pairs for 3 epochs.

Our Method (Progressive Multi-Stage SFT). We implement two variants of our proposed progressive training strategy:

- **Full Three-Stage Training**: this follows the complete curriculum described in Section 4.2. For a given base model, we first train it on the theorem-proof pairs for 3 epochs, then on the theorem-sketch-proof for another 3 epochs and finally on the full hierarchical data for a final 3 epochs.

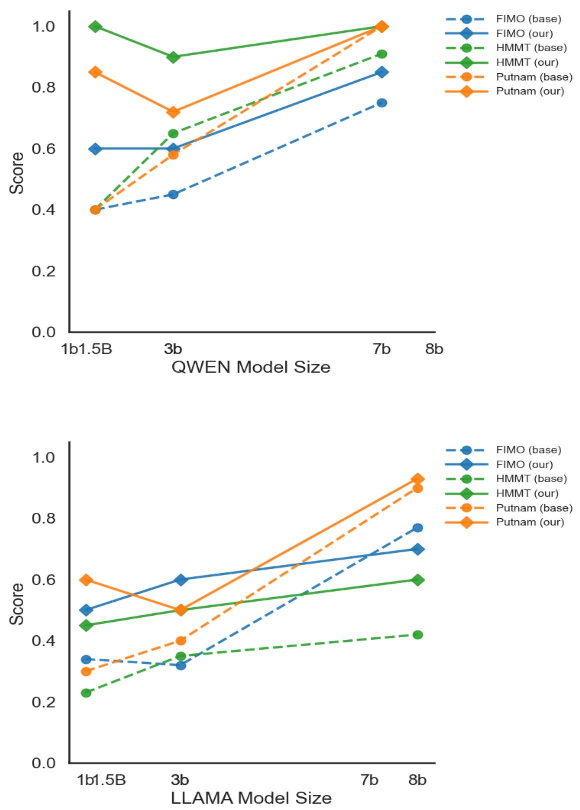

Figure 4: Max scores in evaluation

- **Two-Stage Training**: This is a streamlined variant where we skip the intermediate theorem-sketch-proof stage. The rationale for this design is that the conceptual jump from generating a full proof to generating a proof sketch might be relatively small in terms of reasoning difficulty.

6.1 Main Results

Our framework achieves better performance. The main results are presented in Table 2. These results show that after training for insightful hierarchical reasoning, for both series, the models of different sizes demonstrate superior performance to base models on all three benchmarks. This verifies that, via insightful reasoning, the model's reasoning capability improves. Moreover, Figure 4 shows the max scores for each benchmark, which are pushed higher when involving insight, especially for smaller models like 1.5B backbone. This implies that insightful thinking can help break the reasoning ability ceiling for small models.

6.2 Qualitative Case Study

We provide a case study to show the improved evaluation result for the output of a problem example in HMMT generated by the Qwen2.5-7B

Models	Methods	FIMO	Putnam	HMMT	Avg.
Qwen2.5-1.5B	Base	11.17	25.39	11.25	15.94
	Two-stage	11.81	**28.72**	11.80	17.44
	Three-stage	**12.57**	27.06	**13.68**	**17.77**
Qwen2.5-3B	Base	11.92	31.87	12.69	18.83
	Two-stage	12.85	34.86	11.98	19.90
	Three-stage	**13.34**	**35.69**	**14.06**	**21.03**
Qwen2.5-7B	Base	15.27	36.75	14.82	22.28
	Two-stage	16.33	**43.34**	15.59	25.09
	Three-stage	**18.03**	41.67	**17.78**	**25.83**
Llama3.2-1B	Base	5.04	14.61	4.48	8.04
	Two-stage	4.19	**18.26**	5.78	9.41
	Three-stage	**6.25**	**18.26**	**7.60**	**10.70**
Llama3.2-3B	Base	9.29	23.90	7.04	13.41
	Two-stage	**11.08**	24.73	**9.27**	15.03
	Three-stage	10.72	**26.73**	8.89	**15.45**
Llama3-8B	Base	13.48	37.68	12.25	21.14
	Two-stage	**14.70**	**40.51**	14.15	**23.12**
	Three-stage	14.62	38.51	**14.30**	22.48

Table 2: Evaluation of models trained with our methods and the baseline

model after trained on DeepInsightTheorem against the baseline. The chosen problem is " 2025-02-combinatorics-04 ".

Evaluation of Our Method: we get a score of 0.9 for this problem with a minor completeness issue.

Logical Validity: " `The algorithm is correct and would work, no logical flaw in the structure.` "

Completeness: " `the algorithm is complete, and the answer is stated, but the computation is omitted. It's partial.` "

Clarity: " `The steps are well-explained and clear.` "

Evaluation of the Baseline: The baseline produces a low-quality proof, receiving a score of 0.19.

Logical Validity: " `The proof is logically invalid due to an incorrect reduction to only right and up moves, a flawed bijection in Step 3 and the contradiction between Step 4 and Step 5.` "

Completeness: " `The proof is incomplete as it misses paths using diagonal moves, does not account for variable path lengths, and fails to properly handle the no-revisit constraint.` "

Clarity: " `The proof has a structured outline, but the marking system in Step 3 is ambiguous, and the numerical inconsistency creates confusion.` "

In summary, our methods help the model establish a solid logical ability and cultivate its insight to plan for the reasoning path.

7 Conclusion

In this paper, we considered the structure of proofs in mathematical theorem-proving and pointed out that as thinking bottlenecks the core techniques are hard for LLMs to realize based on fundamental SFT on theorem-proof pairs. Then we constructed a hierarchical dataset and proposed a corresponding progressive variant of SFT to help train LLMs for an insight-induced inference, through which LLMs can improve their mathematical reasoning capability by identifying the core techniques. A comprehensive evaluation via LLM-as-Judge on both proof and insight components of model generation illustrates the effectiveness of our method, achieving superior performance to base theorem-proving patterns on three challenging benchmarks. Throughout these contributions, our method provides a solid and robust foundation for future advancements in automated mathematical theorem proving, especially via natural language, to empower scalable high-level expert-like math reasoning capability in large language models.

Limitations

While our insight-aware hierarchy improves informal theorem proving, several limitations remain. First, our $(q, \{t_i\}, s, p)$ annotations are produced with LLM-assisted decomposition, which may introduce noise; future work will strengthen this pipeline via self-consistency, cross-model agreement, and selective human auditing. Second, our evaluation relies on LLM-as-Judge, which can be prompt-sensitive; we will incorporate multi-judge aggregation and hybrid automatic checks (e.g., structure/consistency and domain-specific symbolic validation when applicable). Finally, our experiments cover a limited set of benchmarks and informal proofs cannot guarantee correctness; we plan broader out-of-distribution evaluations and tighter integration with verification-oriented feedback (e.g., checking key subclaims or partially formalized steps).

Ethical Statement

This work enhances LLM capabilities for informal theorem proving to support research and education, though it does not guarantee formal correctness. Because the model can generate convincing yet potentially erroneous proofs, we recommend expert oversight for high-stakes applications and appropriate safeguards against academic dishonesty. Additionally, as our evaluation relies on LLM judges, results should be interpreted with awareness of potential biases and limitations in detecting subtle mathematical flaws.

References

DeepSeek-AI. 2025. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *CoRR*.

Kefan Dong, Arvind Mahankali, and Tengyu Ma. 2024. Formal theorem proving by rewarding llms to decompose proofs hierarchically. *Preprint*, arXiv:2411.01829.

Google Gemini Team. 2025. Gemini 2.5: Pushing the frontier with advanced reasoning, multimodality, long context, and next generation agentic capabilities. Technical report, Google.

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

Runquan Gui, Zhihai Wang, Jie Wang, Chi Ma, Huiling Zhen, Mingxuan Yuan, Jianye Hao, Defu Lian, Enhong Chen, and Feng Wu. 2025. Hypertree planning: Enhancing llm reasoning via hierarchical thinking. *Preprint*, arXiv:2505.02322.

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

Jiewen Hu, Thomas Zhu, and Sean Welleck. 2025a. minictx: Neural theorem proving with (long-)contexts. *Preprint*, arXiv:2408.03350.

Jinwu Hu, Dongjin Yang, Langyu Bian, Zhiquan Wen, Yufeng Wang, Yaofo Chen, Bin Xiao, Yuanqing Li, and Mingkui Tan. 2025b. Beyond fast and slow: Cognitive-inspired elastic reasoning for large language models. *Preprint*, arXiv:2512.15089.

Haohan Lin, Zhiqing Sun, Sean Welleck, and Yiming Yang. 2025. Lean-star: Learning to interleave thinking and proving. *Preprint*, arXiv:2407.10040.

Chengwu Liu, Jianhao Shen, Huajian Xin, Zhengying Liu, Ye Yuan, Haiming Wang, Wei Ju, Chuanyang Zheng, Yichun Yin, Lin Li, Ming Zhang, , and Qun Liu. 2023a. FIMO: A challenge for- mal dataset for automated theorem proving. *CoRR*.

Chengwu Liu and 1 others. 2023b. Fimo: A challenge formal dataset for automated theorem proving. *arXiv preprint arXiv:2309.04295*.

Pan Lu, Jiayi Sheng, Luna Lyu, Jikai Jin, Tony Xia, Alex Gu, and James Zou. 2025. Solving inequality proofs with large language models. *Preprint*, arXiv:2506.07927.

Gabriel Poesia, David Broman, Nick Haber, and Noah D. Goodman. 2024. Learning formal mathematics from intrinsic motivation. *Preprint*, arXiv:2407.00695.

Haoran Sun and Shaoning Zeng. 2025. Hierarchical memory for high-efficiency long-term reasoning in llm agents. *Preprint*, arXiv:2507.22925.

AIME Dataset. *AIMO Validation AIME Dataset*. Url: https://huggingface.co/datasets/AI-MO/aimo-validation-aime.

Llama Team. 2024. The llama 3 herd of models. Technical report, AI @ Meta.

Qwen Team. 2025. Qwen2.5 technical report. Technical report, Qwen.

Harvard-MIT Mathematics Tournament. 2025. Hmmt dataset and resources.

George Tsoukalas, Jasper Lee, John Jennings, Jimmy Xin, Michelle Ding, Michael Jennings, Ami tayush Thakur, and Swarat Chaudhuri. 2024. Putnambench: Evaluating neural theorem-provers on the putnam mathematical competition. *Advances in Neural Information Processing Systems 38: Annual Conference* *on Neural Information Process- ing Systems 2024, NeurIPS 2024, Vancouver, BC, Canada.*

Guan Wang, Jin Li, Yuhao Sun, Xing Chen, Changling Liu, Yue Wu, Meng Lu, Sen Song, and Yasin Abbasi Yadkori. 2025. Hierarchical reasoning model. *Preprint*, arXiv:2506.21734.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Ed H. Chi, Quoc Le, and Denny Zhou. 2022. Chain of thought prompting elicits reasoning in large language models. *CoRR*, abs/2201.11903.

Sean Welleck, Jiacheng Liu, Ximing Lu, Hannaneh Hajishirzi, and Yejin Choi. 2022. Naturalprover: Grounded mathematical proof generation with language models. *Advances in Neural Information Processing Systems 35: Annual Conference on Neural Information Processing Systems 2022, NeurIPS 2022, New Orleans, LA, USA.*

Ling Yang, Zhaochen Yu, Bin Cui, and Mengdi Wang. 2025. Reasonflux: Hierarchical llm reasoning via scaling thought templates. *Preprint*, arXiv:2502.06772.

Ziyu Ye, Jiacheng Chen, Jonathan Light, Yifei Wang, Jiankai Sun, Guohao Li, Mac Schwager, Philip Torr, Yuxin Chen, Kaiyu Yang, Yisong Yue, and Ziniu Hu. 2024. Reasoning in reasoning: A hierarchical framework for (better and faster) neural theorem proving. In *The 4th Workshop on Mathematical Reasoning and AI at NeurIPS'24.*

Ziyin Zhang, Jiahao Xu, Zhiwei He, Tian Liang, Qiuzhi Liu, Yansi Li, Linfeng Song, Zhenwen Liang, Zhuosheng Zhang, Rui Wang, Zhaopeng Tu, Haitao Mi, and Dong Yu. 2025a. DEEPTHEOREM: Advancing llm reasoning for theorem proving through natural language and reinforcement learning.

Ziyin Zhang and 1 others. 2025b. Deeptheorem: Advancing llm reasoning for theorem proving through natural language and reinforcement learning. *arXiv preprint arXiv:2505.23754.*

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 (ICLR).* ArXiv:2109.00110.

Pei Zhou, Jay Pujara, Xiang Ren, Xinyun Chen, Heng-Tze Cheng, Quoc V. Le, Ed H. Chi, Denny Zhou, Swaroop Mishra, and Huaixiu Steven Zheng. 2024. Self-discover: Large language models self-compose reasoning structures. *Preprint*, arXiv:2402.03620.

A Related Work

A.1 Theorem Proving

Formal theorem proving. Formal automated theorem proving (ATP) relies on proof assistants such as Lean, Coq, and Isabelle (Zheng et al., 2022, Liu et al., 2023a, Tsoukalas et al., 2024), where correctness is enforced by a strict formal system. Recent work integrates large language models with formal proof environments to combine natural-language generation with machine-checkable verification (Gloeckle et al. (2024), Hu et al. (2025a), Lin et al. (2025), Poesia et al. (2024)). Representative benchmarks include miniF2F for cross-system olympiad-level evaluation (Zheng et al., 2022), as well as more challenging datasets that target IMO-/Putnam-style mathematics (Liu et al., 2023b; Tsoukalas et al., 2024). While formal approaches offer strong verifiability, they often face (i) a substantial gap between human mathematical exposition and formal languages, and (ii) a large proof search space that benefits from stronger decomposition, planning, and retrieval.

Informal theorem proving. Informal theorem proving generates proofs directly in natural language and standard mathematical notation (e.g., LaTeX), which aligns well with typical LLM pretraining. NaturalProofs constructs natural-language theorem–proof corpora (Welleck et al., 2022), and NaturalProver studies grounded proof generation under reference/retrieval settings (Welleck et al., 2022). Tencent's DeepTheorem further advances IMO-level informal proving with large-scale data and reinforcement-learning-style training recipes (Zhang et al., 2025b). Lu et al. (2025) specially focuses on inequality proof. Despite strong progress, many systems remain largely end-to-end, which can make it difficult to explicitly surface and control the key ideas (e.g., core techniques) driving a proof.

A.2 Hierarchical Reasoning

Hierarchical reasoning for LLMs decomposes complex problems into manageable sub-problems, enhancing reasoning accuracy and efficiency in complex scenarios. Approach in HyperTree Planning (Gui et al., 2025) constructs a structured, high-level outline before generating details. ReasonFlux (Yang et al., 2025) moves beyond raw text generation by scaling "thought templates". SELF-DISCOVER (Zhou et al., 2024) enables LLMs to self-compose reasoning structures from atomic modules. CoGer (Hu et al., 2025b) draws on Bloom's taxonomy to classify query complexity into four levels and trains an agent via reinforcement learning to dynamically select appropriate reasoning strategies. Another line of work lies in modifying model architectures to support hierarchical processing such as Wang et al. (2025), Sun and Zeng (2025).

Some works focus on the hierarchical reasoning for theorem proving. For example, Dong et al. (2024) proposes a reinforcement learning based training algorithm that incentivizes LLMs to hierarchically decompose theorems into lemmas. Ye et al. (2024) decomposes complex theorem-proving tasks into small, achievable subgoals to abstract formal proof steps.

The key point in these works basically consists in a physical decomposition of a complex problem, which is equivalent to the proof sketch component (Section 4.1) in our work. Our hierarchical method identifies core techniques, which stands in one higher level platform over sketch. The most similar work to ours is Reason-Flux (Yang et al., 2025). However, it depends on a construction of template library and retrieving it for theorem-proving, which does not equip the LLM itself with a real hierarchical thinking capability. In our work, the LLM has been trained for cultivating insightful thinking, which essentially empowers the LLM with high-level thinking pattern.

B Proof Details in Section 3.1

Recall that $\eta_{\text{TP}}(q)$ is the probability of π_{TP} generating a valid proof of question q. The general form of a proof is (1) and denote ordered sequence $\{t_i\}$, $\{r_j\}$ as t and r respectively. Define $\varphi(t)$ to be the length of t. Then, by Bayes' rule, we have

$$\begin{aligned}
\eta_{\text{TP}}(q) &= \sum_{p\in\mathcal{P}(q)} \pi_{\text{TP}}(p|q) \\
&= \sum_{p\in\mathcal{P}(q)} \prod_i \left(\pi_{\text{TP}}(r_i|s_{i-1},q)\pi_{\text{TP}}(t_i|r_i,s_{i-1},q)\right) \\
&= \sum_{p\in\mathcal{P}(q)} \prod_i \pi_{\text{TP}}(r_i,t_i|s_{i-1},q) \\
&= \sum_{p\in\mathcal{P}(q)} \prod_i \left(\pi_{\text{TP}}(t_i|s_{i-1},q)\pi_{\text{TP}}(r_i|t_i,s_{i-1},q)\right) \qquad (6) \\
&\leq \sum_{p\in\mathcal{P}(q)} \delta^{\varphi(t)}. \qquad (7)
\end{aligned}$$

Note that the sum of (6) and (7) are both actually over r such that r is the basic mathematical reasoning component of a valid proof with the corresponding core technique component t. Moreover, the (7) is obtained by

$$\begin{aligned}
&\pi_{\text{TP}}(t_i|s_{i-1},q) \\
&= \sum_{r_i} \pi_{\text{TP}}(t_i|r_i,s_{i-1},q)\pi_{\text{TP}}(r_i|s_{i-1},q) \\
&\leq \delta, \qquad (8)
\end{aligned}$$

where (8) is obtained by the assumption (2). Note the prefix of t_i is not s_{i-1} but $[s_{i-1}, r_i]$.

C Proof Details in Section 3.2

Since our model will go through two processes learning (Section 3.2), we assume that, after training on the base theorem-proof pairs, for a given question q, we have $\pi_\theta(p|q) > 1-\delta_1$ for each $p \in \mathcal{P}(q)$ and some $\epsilon > 0$. After training on our full hierarchical data, we further assume $\pi_\theta(t|q) > 1-\delta_2$ for $t \in \mathcal{T}(q)$ and allow a small range of loss on basic math reasoning: $\pi_\theta(p|q) > 1-\delta_1-\epsilon$ that may be caused by continued learning. Then we have

$$\begin{aligned}
\eta_\theta(q) &= \sum_{p\in\mathcal{P}(q)} \pi_\theta(p|q) \\
&= \sum_{p\in\mathcal{P}(q)} \pi_\theta(t|q) \prod_i \pi_\theta(r_i|r^{\leq i-1},t,q) \qquad (9) \\
&\geq \sum_{p\in\mathcal{P}(q)} (1-\delta_2)(1-\delta_1-\epsilon)^{\varphi(t)}. \qquad (10)
\end{aligned}$$

D Prompt Templates

We group the whole prompts in Figure 8.

E Additional Experimental Results

E.1 Experiments in Section 2.2

We conduct an experiments on how insightful the recent advanced commercial LLMs can be. We choose Gemini2.5-Flash and Deepseek R1 to generate insights of several problems. The problems are chosen from four categories including algebra, analysis, combinatorics and number theory, each contains 10 mathematical competition level questions from Putnam and FIMO dataset. The insight generation prompt is listed in Appendix D titled 'Insightfulness Evaluation'. Then we evaluate these generated insight by o3-mini through the prompt titled 'Insight Evaluation'. The results show that most of the insights are basically listing the general standard techniques according to the conditions in questions and can not see through the essential approaches. See a randomly chosen evaluation result of the following question. The evaluations for both models are shown in Figure 6 and Figure 7. **Question** Let $\mathbb{R}^+$be the set of positive real numbers. Determine all functions $f : \mathbb{R}^+ \to \mathbb{R}^+$ such that, for all positive real numbers x and y,

$$f(x + f(xy)) + y = f(x)f(y) + 1,$$

the final answer is $f(x) = x + 1$.

Insights generated by Deepseek R1 The functional equation involves terms like $f(x + f(xy))$ and $f(x)f(y) + 1$, suggesting the use of **substitution techniques** such as setting $y = 1$ or $x = 1$ to simplify and obtain relations involving $f(1)$. A potential approach is to **explore injectivity/surjectivity** by fixing one variable and comparing expressions, then using **Cauchy-type ideas** after establishing f is linear or affine. The structure $f(x + f(xy)) - f(x)f(y) = 1 - y$ hints at eventually proving f is of the form $ax + b$ and then determining constants via substitution into the original equation.

Insights generated by Gemini2.5-Flash The presence of $f(x+f(xy))$ and $f(x)f(y)$ suggests looking for properties like injectivity, surjectivity, or particular values of f. Substituting specific values (e.g., $x = 1, y = 1$) or analyzing the fixed points of f might be crucial. The term $f(xy)$ combined with $f(x)f(y)$ often points to multiplicative properties or comparison with Cauchy's functional equation variants.

We see that most of the insights generated by both Gemini2.5-Flash and Deepseek-r1 for this question are only standard analysis without hitting the essence. The accuracy of Gemini generated insights is even worse, being considered as " simple scratch ".

E.2 Main Experiments in Section 6

Here we provide some main parameters setting for our experiments. For both model series, we use the learning rate of 2e-5, and training batch size 256. The maximal sequence length is set to be 4096 for all experiments on base dataset, and 8192 for all on DeepInsiightTheorem.

Benchmark	Num.
FIMO	71
Putnam	166
HMMT	76

Table 3: The number of questions in each benchmark

F Data Construction

As we discussed before, the construction of our data is based on the DeepTheorem (Zhang et al., 2025a) by annotating each piece of data with the assist of Deepseek-R1 to help generate the core techniques and corresponding proof sketches. After the generation process, we have a filtration to remove some data that failed to be annotated due to some reasons like temporary API calling failure or annotated with an undesired structure. Also we notice some repeated questions in the base dataset hence we also remove those redundant ones.

Finally, we collect 10,4751 pieces of hierarchical data with rigorous structure as shown in Figure 9. The distribution of each class of techniques are summarized in Figure 5.

Maximizing Data Utility. A fundamental principle of DeepInsightTheorem is its self-contained nature. The creation of the hierarchical structure $(q, \{t_i\}, s, p)$ is achieved solely through the analysis of the information already embedded within the original proofs. By doing so, we increase the informational density of each training example, offering a powerful pathway to improve data efficiency when scaling high-quality mathematical proof data.

G Evaluation

The following Table 3 is a simple statistic on the number of questions in three benchmarks we use for evaluation. Note that in Zhang et al. (2025a), they manually expand each benchmark by some variants of each question. Here we do not need such a design.

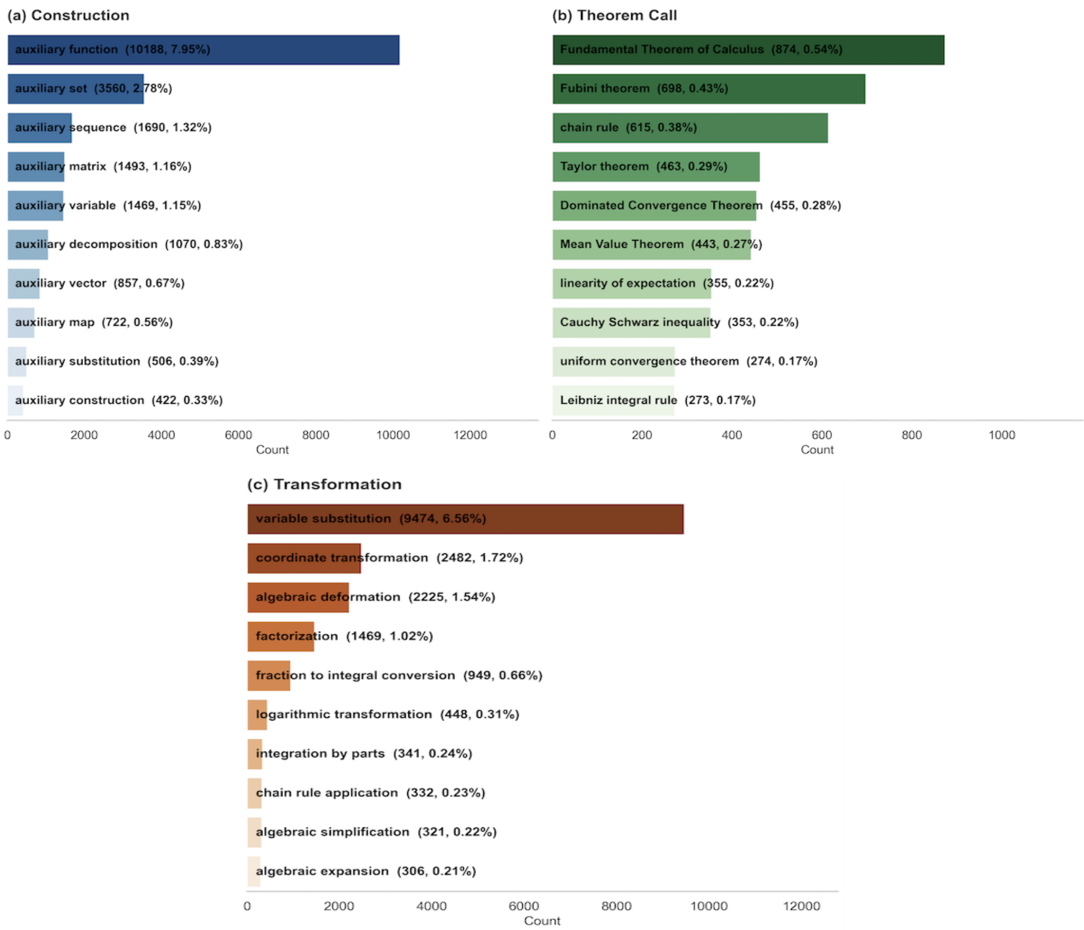

Figure 5: The top technique distribution of each class in DeepInsightTheorem

H Statement on the Use of AI-Assisted Tools

We used AI-assisted tools to support the writing and editing of this manuscript. Specifically, these tools were employed to help refine grammar, improve readability, and enhance the clarity and consistency of the presentation. All technical content—including the problem formulation, algorithm design, theoretical analysis, experimental setup, results, and conclusions—was developed and verified by the authors. The authors reviewed, validated, and edited all AI-assisted outputs to ensure correctness, originality, and compliance with the venue's policies.

O3-mini's Evaluation-Deepseek-r1

1. **mixed**
2. **mixed**
3. **incomplete**
Below is an explanation for each answer:
1. **Mixed:**
The insight mentions several techniques—substitution (setting \(y=1\) or \(x=1\)), exploring injectivity/ surjectivity, and even hinting at Cauchy-type ideas as a route to establishing linearity. However, these ideas remain quite standard for this type of functional equation problem. They are useful, but the answer doesn't pinpoint any single, non-trivial transformation or method that clearly stands out as a breakthrough insight; it rather relies on a collection of standard tactics. Thus, while not entirely shallow, the ideas include both non-shallow observations and standard, almost routine deductions.
2. **Mixed:**
The insight does provide some accurate general directions, which are key in many functional equation proofs. However, the description remains somewhat vague and does not detail enough concrete, non-trivial steps. This mix of accurate generalizations and lack of detailed construction leads to a "mixed" evaluation.
3. **Incomplete:**
While the insight mentions common techniques, it does not capture all potential core ideas needed to solve the problem. There could be additional subtleties—such as a rigorous step-by-step approach to prove that \(f\) must be affine or specific manipulations that pivot the equation to reveal the structure \(f(x) = x+1\). Therefore, the core techniques mentioned are not fully comprehensive to cover all aspects of the solution; they provide an initial guide but remain incomplete.

Figure 6: o3-mini's evaluation response for the insight generated by Deepseek R1

O3-mini's Evaluation-Gemini2.5-Flash

1. **mixed**
2. **simple scratch**
3. **incomplete**
Below is the explanation for each:
1. **Mixed:**
The insight mentions exploring properties like injectivity, surjectivity, particular substitutions (e.g., \(x=1\), \(y=1\)), and fixed points, and notes that the structure of the equation suggests multiplicative features. While these are useful starting points, they represent standard techniques in functional equation problems. They do hint at a deeper structure but do not provide a concrete key step that addresses the heart of obtaining \(f(x)=x+1\). Thus, there exists a mix of non-trivial ideas combined with basic, common observations.
2. **Simple Scratch:**
The insight lists several exploratory steps but does so in a rather vague manner without a precise roadmap tailored to this particular question. The ideas are on the right track but lack the detail needed for a detailed, step-by-step core technique application.
3. **Incomplete:**
While the insight addresses several common techniques for approaching functional equations, it does not cover all the essential steps needed to rigorously derive \(f(x)=x+1\). Additional deeper reasoning, such as deducing specific values (like evaluating \(f(1)\)) and showing uniqueness or employing clever substitutions that eventually simplify the functional relationship, is necessary. Thus, the list of techniques provided is not comprehensive enough to fully solve the problem without filling in significant logical and algebraic deductions.

Figure 7: o3-mini's evaluation response for the insight generated by Gemini2.5-Flash

Insight Generation

Now you are tested the insight this math question:
{question}
From now on you can not go on thinking and expanding the proof step by step. Instead, try to infer the most potential core techniques that would be used in this question by ONLY analyzing the conditions in the question. You can only have a short and quick guess without detailed and long deduction.
The core techniques may refer to specific mathematical construction, use of known theorem or existing results or smart and subtle mathematical transformations, instead of those fundamental logic deduction.
Output: Latex formatted core precise techniques in your prediction for around 3 sentences.

Insightfulness Evaluation

The core techniques in math proof may refer to specific mathematical construction, use of known theorem or existed results or smart and subtle mathematical transformations for particular question, instead of those fundamental mathematical and logic deduction details. Now you have the following question:
{}.
And a given insight which contains the prediction of core techniques to this question:
{}
Now you need to review and evaluate this insight from the following aspects:
1. see if there exists an idea in this insight that is highly key to the solution, which in general is likely to be found or realized after several thinking steps, or the other steps in the solution are far easier after dealing with this core idea. Note that such idea can not vaguely hints without providing concrete methods and identifying the core non-trivial step. And hence determine whether the insight is shallow quick guess or deep identification.
2. see if the core ideas described in the insight is accurate enough. Check if the idea gives the precise techniques key and adapted particularly to the question. It does not need to be containing details but accurately describe mentioned techniques. And hence determine whether the idea is a simple scratch or an accurate expression.
3. see if the core techniques mentioned in this insight are all core techniques for the question.
check whether the question can be solved by filling basic mathematical and logical deductions ONLY under the core techniques in the insight. And hence determine whether the insight is comprehensive or incomplete.

--output:
For 1, if all ideas in the insight are not shallow quick guess, and there are no flaws in the whole insight, then output 'deep identification'. If there are also some ideas that are shallow quick guess, or even there are some flaws in this insight, output 'mixed'. Note that mixed means there indeed exist non-shallow ideas, but also other standard observations. Otherwise, if there are all shallow quick guess or just spread out standard general techniques in the insight without an accurate orientation, then output 'shallow quick guess'.
For 2, if all ideas in the insight are satisfied, then output 'detailed expression'. If all are not satisfied, output 'simple scratch'. otherwise output 'mixed'.
For 3, if all ideas in the insight are enough core ideas for the question, then output 'comprehensive'. If not, output 'incomplete'.
format: 1. 'deep identification'/ 'shallow quick guess'/'mixed'
2. 'detailed expression'/'simple scratch'/'mixed'
3. 'comprehensive'/'incomplete'/'mixed'

Data Construction

Analyze the mathematical problem and solution below:
Problem:
{question}
Solution:
{response}
Perform these tasks:
1. First identify 1-3 core mathematical techniques used in the solution by considering if there are some specific constructions, theorem or existed results calling
and smart mathematical transformations, where the smart transformation may not be known results and are subtle and hard to note.
Note that the core mathematical techniques are not those basic logic deductions. They need to be crucial to the solution.
Then Write the analysis for each technique on how a person can realize such technique when reading the question. The analysis should be from several aspects, including how the problem settings or assumptions suggest this technique,
how it might be potentially useful to prove the result and why it might be crucial to the whole proof.
The analysis should be like an insightful and experienced math professor's thoughts when he is trying to solve the question. The whole analysis should start with 'Let's analyze the conditions in this question.'
The analysis contains 2-3 sentences for each technique. The analysis should avoid mentioning the solution. Make sure the logic of the analysis is coherent. After analysis, write the extracted techniques following the analysis through 'Therefore, the potential techniques are summarized as...', with three tags:

<construction>: identify the specific construction used in the solution if any;
<theorem call>: specify the theorem or any existed results used in the solution if any;
<transformation>: specify the smart mathematical transformations used in the solution if any.
If there are no such techniques, just specify 'no' after the tag, but you must write the tag <construction>, <theorem call> and <transformation> even there are no such techniques.
Do not write one technique under two tags both.
The whole technique analysis, e.g., the analysis and technique extraction, should be with LaTeX inline math ($...$) where appropriate.
2. Create a proof sketch integrating these techniques analyzed from task 1, which serves as a high-level proof organization:
- Format: Numbered steps in LaTeX
- Each step: 1 sentence with key mathematical reasoning
- Include essential formulas in math mode
- Example:'\\begin{{enumerate}}\\item Assume P is countable: $P = \\{{x_1, x_2, \\...\\}}$\\end{{enumerate}}'
3. Based on the original given solution, improve the solution by elaborating each step in the proof sketch, making it well-organized.
4. Output:
- mathematical techniques: string (LaTeX formatted)
- proof sketch: string (LaTeX enumerated steps)
- solution: string (LaTeX formatted)
The mathematical techniques are enclosed within <tech></tech>, the proof sketch is enclosed within <sketch></sketch> and the solution is enclosed within <proof></proof>, respectively,
i.e., <tech> mathematical technique analysis here </tech> <sketch> proof sketch here </sketch> <proof> solution here </proof>

Proof Evaluation

The following question asks to prove a statement.
The question:
{question}
The test subject's solution:
{response}
Your task is to evaluate the proof's quality and assign a score from 0 to 1 based on three criteria: logical validity (40%), completeness (30%), and clarity (30%).
Instructions:
1. Analyze the proof step by step.
2. For each criterion:
- Logical Validity: Check if each step follows logically from the previous one. Flag any logical errors.
- Completeness: Verify if all necessary cases and steps are included to prove the theorem.
- Clarity: Assess if the proof is clear, unambiguous, and well-explained.
3. Assign a sub-score (0 to 1) for each criterion and compute the total score using the weights: (0.4 × validity) + (0.3 × completeness) + (0.3 × clarity).
4. Provide a brief explanation (2-3 sentences) summarizing any errors or issues and justifying the score.

Output:
- Your total score: float;
- Your sub-scores and corresponding brief explanation:
  - Sub-score: float;
  - Brief explanation: string (LaTeX formatted)
The total score is enclosed in <score></score>, and the sub-scores with corresponding explanation are enclosed in <exp></exp>, e.g.,
- Format:
'<score>
your final scores here. Just write a single number here.
</score>
<exp>
"validity": your sub-score for validity here
explanation: your explanation for validity score here
"completeness": your sub-score for completeness here
explanation: your explanation for completeness score here
"clarity": your sub-score for clarity here
explanation: your explanation for clarity score here
</exp>'

Figure 8: The above is a list of prompts

Question:
Let X be a non-negative random variable and $g : [0, \infty) \to [0, \infty)$ be a monotone nondecreasing function such that $g(x) \to \infty$ as $x \to \infty$. If $g(X)$ is integrable, i.e., $E[g(X)] < \infty$, prove that $\lim_{x\to\infty} g(x)E[X > x] = 0$.
Response:
<tech>
Let's analyze the conditions in this question. The monotonicity and divergence of g naturally suggest linking the tail event $\{X > x\}$ to the growth of $g(X)$, forming a crucial inclusion that leverages g's behavior. The integrability of $g(X)$ prompts Markov's inequality to bound the tail probability by an expectation, exploiting the non-negativity and finite mean. The pointwise convergence to zero, combined with the integrable bound $g(X)$, calls for the Dominated Convergence Theorem to justify limit-interchange, which is indispensable since the expectation structure captures the asymptotic decay. Therefore, the potential techniques are summarized as:
<construction>: Event inclusion $\{X > x\} \subseteq \{g(X) \geq g(x)\}$ via monotonicity of g
<theorem call>: Markov's inequality and Dominated Convergence Theorem
<transformation>: no
</tech>
<sketch>

1. Use the monotonicity and divergence of g to establish $\{X > x\} \subseteq \{g(X) \geq g(x)\}$, implying $P(X > x) \leq P(g(X) \geq g(x))$.

2. Apply Markov's inequality to the nonnegative integrable $g(X)$, yielding $g(x)P(X > x) \leq E[g(X)\mathbf{1}_{\{g(X)\geq g(x)\}}$.
3. Observe pointwise convergence $g(X)\mathbf{1}_{\{g(X)\geq g(x)\}} \to 0$ due to $g(x) \to \infty$, and dominate by $g(X)$.
4. Invoke the Dominated Convergence Theorem to show $E[g(X)\mathbf{1}_{\{g(X)\geq g(x)\}} \to 0$.
5. Conclude $\lim_{x\to\infty} g(x)P(X > x) = 0$ via the squeeze theorem on $0 \leq g(x)P(X > x) \leq E[g(X)\mathbf{1}_{\{g(X)\geq g(x)\}}$.

</sketch>
<proof>
Since g is monotone nondecreasing, for any $x \geq 0$, the condition $X > x$ implies $g(X) \geq g(x)$. This gives the event inclusion:

$$\{X > x\} \subseteq \{g(X) \geq g(x)\},$$

which, by monotonicity of probability, yields:

$$P(X > x) \leq P(g(X) \geq g(x)).$$

As $g(X)$ is nonnegative and $E[g(X)] < \infty$, Markov's inequality for $a = g(x) > 0$ (for large x since $g(x) \to \infty$) states:

$$P(g(X) \geq g(x)) \leq \frac{E[g(X)\mathbf{1}_{\{g(X)\geq g(x)\}}]}{g(x)}.$$

Multiplying by $g(x)$ and using the earlier inequality, we obtain:

$$g(x)P(X > x) \leq g(x)P(g(X) \geq g(x)) \leq E[g(X)\mathbf{1}_{\{g(X)\geq g(x)\}}].$$

Fix $\omega \in \Omega$. As $x \to \infty$, $g(x) \to \infty$ so $g(x) > g(X(\omega))$ for large enough x, forcing $\mathbf{1}_{\{g(X(\omega))\geq g(x)\}} = 0$. Thus:

$$\lim_{x\to\infty} g(X(\omega))\mathbf{1}_{\{g(X(\omega))\geq g(x)\}} = 0, \quad \forall\omega.$$

Moreover, $0 \leq g(X)\mathbf{1}_{\{g(X)\geq g(x)\}} \leq g(X)$, and $E[g(X)] < \infty$ by hypothesis. By the Dominated Convergence Theorem:

$$\lim_{x\to\infty} E[g(X)\mathbf{1}_{\{g(X)\geq g(x)\}}] = E\left[\lim_{x\to\infty} g(X)\mathbf{1}_{\{g(X)\geq g(x)\}}\right] = E[0] = 0.$$

Finally, sandwiching with:

$$0 \leq g(x)P(X > x) \leq E[g(X)\mathbf{1}_{\{g(X)\geq g(x)\}}] \xrightarrow{x\to\infty} 0,$$

we conclude $\lim_{x\to\infty} g(x)P(X > x) = 0$ as required.
</proof>

Figure 9: This is an example of DeepInsightTheorem