StepFun-Formalizer: Unlocking the Autoformalization Potential of LLMs Through Knowledge-Reasoning Fusion

Yutong Wu[1, 2], **Di Huang**[1], **Ruosi Wan**[3], **Yue Peng**[3], **Shijie Shang**[3], **Chenrui Cao**[1, 4], **Lei Qi**[1, 4], **Rui Zhang**[1], **Xishan Zhang**[1], **Zidong Du**[1], **Jie Yan**[3], **Xing Hu**[1*]

[1]State Key Lab of Processors, Institute of Computing Technology, CAS
[2]University of Chinese Academy of Sciences
[3]StepFun Inc.
[4]University of Science and Technology of China
wuyutong22s@ict.ac.cn

Abstract

Autoformalization aims to translate natural-language mathematical statements into a formal language. While LLMs have accelerated progress in this area, existing methods still suffer from low accuracy. We identify two key abilities for effective autoformalization: comprehensive mastery of formal-language domain knowledge, and reasoning capability of natural language problem understanding and informal-formal alignment. Without the former, a model cannot identify the correct formal objects; without the latter, it struggles to interpret real-world contexts and map them precisely into formal expressions. To address these gaps, we introduce ThinkingF, a data synthesis and training pipeline that improves both abilities. First, we construct two datasets: one by distilling and selecting large-scale examples rich in formal knowledge, and another by generating informal-to-formal reasoning trajectories guided by expert-designed templates. We then apply SFT and RLVR with these datasets to further fuse and refine the two abilities. The resulting 7B and 32B models exhibit both comprehensive formal knowledge and strong informal-to-formal reasoning. Notably, StepFun-Formalizer-32B achieves SOTA BEq@1 scores of 40.5% on FormalMATH-Lite and 26.7% on ProverBench, surpassing all prior general-purpose and specialized models.

Code — https://github.com/stepfun-ai/StepFun-Formalizer

Models — https://huggingface.co/stepfun-ai/StepFun-Formalizer-32B

Extended version — https://arxiv.org/abs/2508.04440

1 Introduction

Autoformalization aims to translate natural-language mathematical statements into formally verifiable statements in formal languages such as Lean (Moura and Ullrich 2021), Coq (Barras et al. 1999), and Isabelle (Paulson 1994). With recent advances in automated theorem proving (Li et al. 2024) and formal verification (Beg, O'Donoghue, and Monahan 2025), it has garnered growing interest (Weng et al. 2025) for underpinning data synthesis for theorem provers (Xin et al. 2024a,b; Lin et al. 2025; Zhang et al. 2025; Wang et al. 2025; Ren et al. 2025; Ji et al. 2025), the validation of informal mathematical reasoning (Zhou et al. 2024), and the generation of verifiable code (Lin et al. 2024; Thakur et al. 2025; Miranda et al. 2025).

The current mainstream approaches for autoformalization involve employing an LLM to translate informal mathematical problems into their corresponding formal statements. These approaches can be categorized into two types: (1) Directly distilling or prompting general-purpose LLMs, such as FormalMATH (Yu et al. 2025b), which generates autoformalization training data using GPT-4, and FMC (Xie et al. 2025), which directly employs DeepSeek-R1 (DeepSeek-AI 2025) as the translation model. (2) Training a specialized model for the autoformalization task from scratch, starting with existing manually annotated informal-formal pairs and followed by supervision from human experts, such as Lean Workbook (Ying et al. 2025) and Kimina-Autoformalizer (Wang et al. 2025). However, these two types of methods suffer from low accuracy and require substantial human effort to check and revise the human-generated formal statements (Yu et al. 2025b; Zhang, Valentino, and Freitas 2025).

We identify two key abilities for effective autoformalization models: **(1) Comprehensive mastery of domain knowledge in formal language.** A model must know how to express every mathematical concept in the target formal language. For instance, without familiarity with the Lean 4 definition of Euler's totient function, it cannot formalize related problems (see Appendix E.1 in the FormalMATH paper (Yu et al. 2025b)). Because formal data are scarce in general corpus (Xin et al. 2024a) and significant differences exist between versions of formal languages (e.g., Lean 3 and Lean 4 (Community 2023)), general-purpose LLMs often mislearn or overlook crucial language details, limiting their performance (e.g., Claude4-thinking in Figure 1). **(2) Reasoning capability of natural language problem understanding and informal-formal alignment.** Models tackling real-world problems must first grasp the intended meaning, which is something specialized models with annotated data in narrow scenarios struggle with (e.g., Kimina-Autoformalizer in Figure 1). Moreover, informal problem statements often omit details (e.g., type hints) that must be made explicit in formal language for rigorous verification. Overcoming these gaps requires strong informal-to-formal reasoning abilities, as demonstrated by StepFun-Formalizer (Figure 1) and

*Corresponding author

Informal Problem: Victoria wants to order at least 550 donuts from Dunkin' Donuts for the HMMT 2014 November contest. However, donuts only come in multiples of twelve. Assuming every twelve donuts cost $7.49, what is the minimum amount Victoria needs to pay, in dollars? Prove that the answer is: 344.54.

Claude4-thinking (advanced general-purpose models, lack of formal knowledge):
<think>
Let me think about this problem step by step. Victoria wants to order at least 550 donuts. Donuts come in multiples of 12...
</think>

```
theorem b :
  let target_donuts : ℕ := 550
  let donuts_per_dozen : ℕ := 12
  let min_dozens := Nat.ceil_div
target_donuts donuts_per_dozen
  ...
```

✗

Mistakes: Function `Nat.ceil_div` does not exist in Lean 4.

Kimina-Autoformalizer (specialized models, lack of reasoning):

```
theorem a : IsLeast {x | ∃ n : ℕ, x = n *
7.49 ∧ 12 ∣ n ∧ n ≥ 550} 344.54 := by sorry
```

✗

Mistakes: (1) It misunderstands the meaning of dozen (Box). (2) The total payment **x** is not explicitly declared as a **real number**, causing a type error with the value 7.49.

StepFun-Formalizer (ours, formal knowledge + reasoning):
<think>
The problem involves math several concepts:
2. **Cost calculation**: Each pack of 12 donuts costs $7.49. The total cost is calculated by multiplying the number of packs by the cost per pack...
Based on the problem, we need to work with the following mathematical objects:
3. **Cost**: **This is a real number** ($\mathbb{R}$), calculated as 7.49 × n.
We need to find the minimum cost...
Several potential issues might arise when formalizing this problem in Lean:
1. **Type coercion**: For instance, 7.49 is a real number, but we're multiplying it by a natural number n. Lean will handle this through **automatic coercion**, but we should be aware of it.
...
</think>

```
theorem c : IsLeast {x : ℝ | ∃ n : ℕ, x =
 n * 7.49 ∧ 12 * n ≥ 550} 344.54 := by
sorry
```

✓

Figure 1: A case study to demonstrate the impact of formal knowledge and informal-to-formal reasoning capability on autoformalization models. It shows that general-purpose models without formal knowledge make mistakes in code implementation, while specialized ones without reasoning capability struggle with problem understanding and informal-formal alignment. StepFun-Formalizer improves autoformalization performance by combining these two capabilities.

earlier studies (Liu et al. 2025b; Xuejun et al. 2025; Xie et al. 2025). Quantitatively, we leveraged GPT-4o (OpenAI 2024a) to classify errors in roughly 10K generated outputs and had human experts annotate a 100-example subset from two autoformalization models (Table 1). The results show that Kimina-Autoformalizer suffers a high rate of these two types of errors as an evidence of weak reasoning, whereas our model alleviates these problems.

To equip the model with these two abilities, we introduce ThinkingF, the first data synthesis and training pipeline for the autoformalization model with both domain knowledge and informal-to-formal reasoning capability. We implement this through two key components: (1) large-scale distillation and selection of data with formal knowledge from specialized models, and (2) informal-to-formal reasoning trajectories synthesis guided by an expert-designed autoformalization template.

Specifically, we first construct two datasets to respectively supplement domain knowledge and enhance the model's reasoning ability. For domain knowledge, we collect informal-formal data pairs by employing a specialized model (e.g., Kimina-Autoformalizer) to translate a large number of natural language mathematical problems in public datasets (e.g., NuminaMath-1.5) into formal statements, followed by majority voting and an LLM-based judge to ensure data quality. For reasoning ability, we propose a reasoning template that involves problem decomposition and mathematical object mapping. With this template, we leverage a strong instruction-following model (e.g., Claude 3.7 Sonnet) to synthesize the reasoning process for each informal-formal data pair in existing human-annotated datasets. Next, we use the two datasets to perform supervised fine-tuning on a general-purpose LLM with strong informal mathematical and coding capabilities (e.g. DeepSeek-R1-Distill-Qwen), thereby integrating domain knowledge and informal-to-formal reasoning ability into a unified model. Finally, we apply reinforcement learning to the fine-tuned model to further promote the fusion of the two capabilities, using the BEq equivalence verification (Liu et al. 2025b) as a verifiable reward (RLVR). The overview of ThinkingF is shown in Figure 2.

Based on this pipeline, we develop two sizes of autoformalization LLMs, StepFun-Formalizer-7B and StepFun-

Evaluation	Model	Correct	NLU	IFM	Other
GPT-4o	Kimina	33.7	33.0	28.1	5.2
	StepFun (ours)	42.9	26.0	27.0	4.1
Human	Kimina	35.7	35.7	24.3	4.3
	StepFun (ours)	50.0	26.8	16.1	7.1

Table 1: Categorical analysis for errors in autoformalization model. "Kimina" refers to Kimina-Autoformalizer. It illustrates the proportion (%) of two error types in autoformalization, both of which are mitigated in StepFun-Formalizer. NLU: Natural Language Understanding. IFM: Informal-Formal Misalignment.

Formalizer-32B. We evaluate them using the BEq verification on mainstream benchmarks, including FormalMATH-Lite (Yu et al. 2025b), ProverBench (Ren et al. 2025) and CombiBench (Liu et al. 2025a). Specifically, StepFun-Formalizer-32B achieves 40.5% on FormalMATH-Lite and 26.7% on ProverBench in BEq@1 score (Liu et al. 2025b), setting new SOTA results among both specialized and general-purpose models.

2 Related Work

Large Language Models for Autoformalization. Autoformalization is the process of converting mathematical expressions from natural language into their formal language representations (Weng et al. 2025). Traditional rule-based autoformalization methods are complex to implement and difficult to generalize (Jiang, Li, and Jamnik 2023). Therefore, we mainly focus on LLM-based methods. Earlier works enhance the autoformalization capability of existing LLMs by in-context learning (Wu et al. 2022), data synthesis with back-translation (Jiang, Li, and Jamnik 2023; Azerbayev et al. 2023; Wu et al. 2025), retrieval-augmented generation (Liu et al. 2025b), natural language inference (Ying et al. 2025), and expert iteration with LLM judgers (Wang et al. 2025). Mathesis (Xuejun et al. 2025) is the first autoformalization model with reinforcement learning in its training process, but it does not perform informal-to-formal reasoning during translation, resulting in informal-formal misalignment. Moreover, since Mathesis's model and evaluation methods are not publicly available, we are unable to make a performance comparison. For other public works, we compare with them in Section 5.

The main difference between our method and the aforementioned works is that we equipped the model with both domain knowledge of formal language and informal-to-formal reasoning capabilities, thereby significantly improving its autoformalization capability.

Reasoning-Enhanced Large Language Models. Inspired by the powerful and effective RL paradigm of general reasoning LLMs like OpenAI-o1 (OpenAI 2024b), DeepSeek-R1 (DeepSeek-AI 2025), and Kimi-K1.5 (Kimi et al. 2025), some previous works attempt to integrate reasoning capabilities from general-purpose LLMs into domain-specific LLMs to enhance their performance in solving complex problems, such as Fin-R1 (Liu et al. 2025c), Table-R1 (Yang et al. 2025), CodeV-R1 (Zhu et al. 2025), and R1-Code-Interpreter (Chen et al. 2025).

Since prior works rely on abundant corpora and general LLMs with rich domain knowledge, they typically distil these models to obtain domain-specific reasoning data, but distillation often limits performance below teacher models. In contrast, we first enhance the general model with formal knowledge, then train for reasoning, achieving performance that matches or exceeds both specialized and general models.

3 Problem Statement

The autoformalization task discussed in this paper can be formulated as follows:

Definition 3.1 (LLM-based Autoformalization)**.** *Given an informal mathematical problem x as the input to an autoformalization model $\mathcal{M}$, its output $\mathcal{M}(x)$ is the corresponding formal statement y with an optional reasoning process. We say that y is a correct formalization of x if and only if:*

(1) y passes the syntax check of the formal language.

(2) y is semantically equivalent to x.

Due to the difficulty of automating the strict judgment of semantic equivalence between a model-generated formal statement and an informal problem, we use human-annotated formal statements as the ground truth. We then assess correctness by checking if the model's formalization is equivalent to the human-annotated ground truth using the proof assistant, as in previous work (Liu et al. 2025b):

Definition 3.2 (Bidirectional Extended Definitional Equivalence, BEq)**.** *Two formal statements y_1 and y_2 are bidirectional extended definitional equivalence (denoted as $y_1 \sim y_2$) if and only if there exists a formal proof that derives y_2 from y_1 using semantics-preserving tactics, and vice versa.*

We use the strictest BEq check, i.e., only call the tactic `exact?` to automatically search for an equivalence proof between two statements. See the extended version of the paper for more details.

4 Methods

The data synthesis and training pipeline of ThinkingF consists of 4 stages (Figure 2). Stages ❶ $\sim$ ❷ are used to construct two datasets that enhance the model's formal knowledge and reasoning capabilities: ❶ **Knowledge Distillation With Selection.** To obtain a formal knowledge dataset, we need a large number of high-quality informal-formal pairs. Therefore, we employ a specialized LLM to translate informal problems into formal ones, followed by selection to improve the data quality. ❷ **Informal-to-formal Reasoning Data Synthesis.** Since there is no reasoning process in previous specialized models and distilling reasoning trajectories from general models yields poor results (see Section 5.4), we design a reasoning template for the autoformalization task. Using this template, we synthesize an autoformalization dataset containing informal-to-formal reasoning processes. Stages ❸ $\sim$ ❹ involve the training process of our model, which is used to endow the model with the two capabilities and further promote their fusion: ❸ **Two-Stage Supervised Fine-tuning.** A general-purpose LLM undergoes supervised fine-tuning (SFT) with the two datasets to obtain a model with both capabilities. ❹ **Reinforcement Learning With Verifiable Reward.** To further facilitate the fusion of knowledge and reasoning, we apply reinforcement learning (RL) to train the fine-tuned model, with BEq checking as a verifiable reward.

4.1 Knowledge Distillation With Selection

Informal Problem Preparation Our distillation pipeline begins with the informal mathematical problems $\{x_i\}$ from an open-source dataset, NuminaMath-1.5. Following Kimina-Autoformalizer (Wang et al. 2025), we first filter the dataset by manual rules, resulting in approximately 256K informal problems.

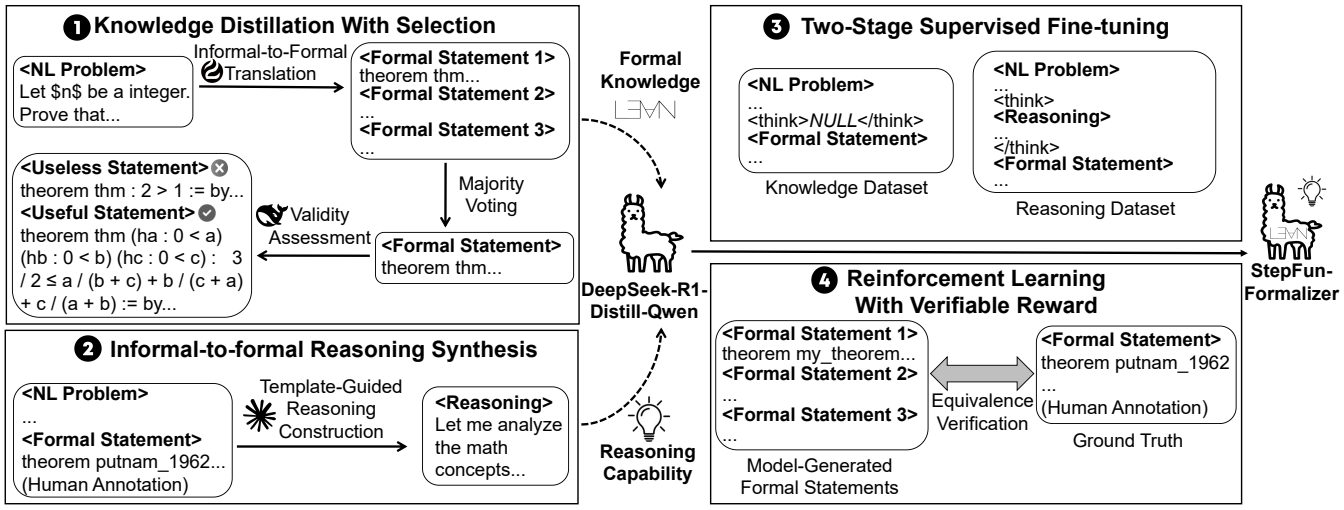

Figure 2: The illustration of ThinkingF method. Our method mainly consists of the construction process for knowledge and reasoning dataset (Section 4.1 and 4.2), and the model training process (Section 4.3 and 4.4).

Formal Statement Generation and Selection We prompt Kimina-Autoformalizer to generate 16 candidate formal statements $\{y_{ij}\}_{j=1}^{16}$ for each informal problem x_i. Next, we perform a three-tier quality selection on the generated formal statements:

(1) Syntax Check. We use the Lean4 REPL to perform syntax checking on the formal statements $\{y_{ij}\}_{j=1}^{16}$, and retain the syntactic correct statements, denoted as $\{y_{ij}^*\}_{j=1}^{m_i}$, where m_i is the number of syntactically correct statements..

(2) Majority Voting. It is observed that majority voting can significantly improve the performance of autoformalization models (see the extended version of the paper), just as it does in coding (Li et al. 2022) and informal math (Wang et al. 2023) tasks.

Specifically, for the syntactically correct formal statements $\{y_{ij}^*\}_{j=1}^{m_i}$, we use BEq verification to partition the formal statements into multiple equivalence classes. Then, one formal statement is randomly selected from the largest group as the optimal formalization y_i^{**} corresponding to the informal problem x_i. Formally:

$$y_i^{**} = \arg\max_{y_{ij}^*} \sum_{k=1}^{m_i} \mathbb{1}(y_{ik}^* \sim y_{ij}^*),$$

where $\mathbb{1}(\cdot)$ is the indicator function, and $\sim$ denotes semantic equivalence. Informal problems and their optimal formalization are collected as a dataset $\{(x_i, y_i^{**})\}$.

(3) Problem Validity Assessment. The selected dataset $\{(x_i, y_i^{**})\}$ is finally evaluated using DeepSeek-V3 (since it is efficient and affordable), removing oversimplified formal statements and those containing inherent contradictions in conditions. We keep approximately 183K informal-formal pairs as the final training data. The ablation study without LLM selection (in the extended version) shows that this step can reduce the amount of training data required while maintaining comparable model performance.

4.2 Informal-to-Formal Reasoning Data Synthesis

Template Design for Reasoning According to the previous error analysis of LLM-based autoformalization, we propose a template-guided reasoning construction framework (Figure 3), which incorporates the human understanding of the autoformalization process to assist the model in generating reasoning trajectories. The framework consists of two parts:

(1) Informal Problem Understanding. Before delving into the details of formal languages, the model needs to deeply understand the natural language problem, including rephrasing the original question, analyzing its high-level logical structure with the decomposition of the mathematical concepts and corresponding objects involved.

(2) Informal-to-Formal Analysis. To bridge the misalignment between natural and formal language, the model should first consider the tricky issues that may arise during formalization. Then, following a divide-and-conquer paradigm (Chen et al. 2024), the model maps the natural language mathematical objects to formal language.

Synthesizing reasoning trajectories upon existing human-annotated data. To maximize the correctness of the model's reasoning, we use informal problems with a human-annotated ground-truth formal statement (denoted as $\{(\hat{x}_i, \hat{y}_i)\}$) as seed data. Following previous work (Wang et al. 2025; Lin et al. 2025), the human-annotated data mainly comes from the matched informal-formal statements in automated theorem-proving datasets (see Section 5.1 for details). With the annotated data pairs $\{(\hat{x}_i, \hat{y}_i)\}$, we prompt Claude 3.7 Sonnet (since it has strong instruction-following capabilities) to generate a reasoning trajectory $\hat{c}_i$ from $\hat{x}_i$ to $\hat{y}_i$ following our reasoning template. In total, we synthesize 5.8K instances of informal-to-formal reasoning data.

4.3 Two-Stage Supervised Fine-tuning

With the knowledge-distilled dataset $\{(x_i, y_i)\}$ and the reasoning dataset $\{(\hat{x}_i, \hat{c}_i, \hat{y}_i)\}$, we conduct two-stage su-

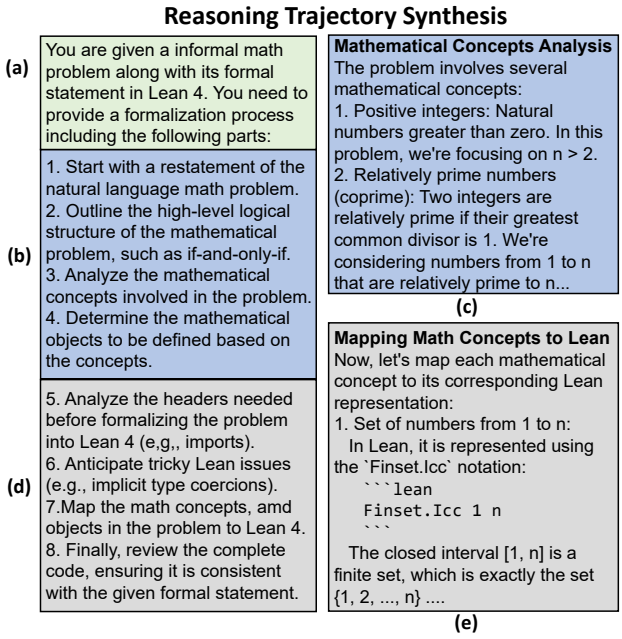

Figure 3: The prompts and examples in the template-guided reasoning construction framework. (a) The task description for autoformalization. (b) Understanding of natural language problems. (c) An example of concept analysis in problem understanding. (d) Analysis of converting informal math objects into formal language. (e) An example of mapping concepts to Lean in informal-to-formal analysis.

pervised fine-tuning (SFT) on DeepSeek-R1-Distill-Qwen, known for its strong reasoning performance in informal mathematics and coding. Specifically, in the first stage of supervised fine-tuning, x_i is used as the input, and special tokens `<think></think>` are inserted before the corresponding output y_i to ensure internal format consistency within the model (Qwen 2025) and maintain its reasoning capability. During the second-stage supervised fine-tuning, we follow the standard format for reasoning models (DeepSeek-AI 2025) by enclosing the reasoning trajectory $\hat{c}_i$ within `<think>` and `</think>` in the output, i.e., `<think>`$\hat{c}_i$`</think>`$\hat{y}_i$. After the two-stage supervised fine-tuning, we obtain a preliminary model, StepFun-Formalizer-SFT, equipped with both formal domain knowledge and informal-to-formal reasoning capability.

4.4 Reinforcement Learning With Verifiable Reward

To further enhance the model's reasoning capability, we perform RL on StepFun-Formalizer-SFT. Due to the lack of high-quality human-annotated data, we use the same set of 5.8K problems $\{(\hat{x}_i, \hat{y}_i)\}$ from the second-stage SFT training for RL training. Despite being used in SFT, continuing RL training on these data still leads to performance improvements (see Section 5.5). The rewards are calculated by performing BEq equivalence verification ($\sim$) between model-generated statements and ground-truth. More formally, the accuracy reward function is defined as follows:

$$R(y_i, \hat{y}_i) = \begin{cases} 1, & \text{if } y_i \sim \hat{y}_i \\ 0, & \text{otherwise} \end{cases}$$

Taking both training speed and performance into account, we choose Group Relative Policy Optimization (GRPO) algorithm (Shao et al. 2024) in reinforcement learning training, which eliminates the value function and estimates the advantage in a group-relative manner. We also incorporate several improvements from Dynamic Sampling Policy Optimization (DAPO) (Yu et al. 2025a), including dynamic sampling and token-level loss.

5 Experiments

In this section, we present the implementation details of ThinkingF (Sec. 5.1). We conduct a series of experiments (settings in Sec. 5.2) to show the performance comparison between StepFun-Formalizer and previous SOTA models (Sec. 5.3), alternative designs in our method (Sec. 5.4), the further analysis of RL training and real-world applications (Sec. 5.5).

5.1 Implementation Details

	Dataset	Size
Training	MiniF2F (Zheng, Han, and Polu 2022)	488
	ProofNet (Azerbayev et al. 2023)	357
	PutnamBench (Tsoukalas et al. 2024)	659
	Compfiles (Renshaw 2024)	115
	FormalMATH-Train (Yu et al. 2025b)	5135
Evaluation	FormalMATH-Lite (Yu et al. 2025b)	425
	ProverBench (Ren et al. 2025)	174
	CombiBench (Liu et al. 2025a)	100

Table 2: Data partitions and sizes. We select a subset of the three most recent datasets for evaluation to ensure fairness, with the remaining datasets for training. Specifically, we use a subset of FormalMATH (FormalMATH-Lite) for evaluation, and the remaining (FormalMATH-Train) for training.

Datasets. The datasets we use for reasoning synthesis, RL training, and evaluation are all collected from automated theorem-proving problem sets (Table 2), which contain informal math problems paired with human-annotated (or model-generated with manually reviewed) formal statements. For problems containing multiple subproblems or lemmas, we retain only the last one. To prevent data contamination, we perform 13-gram decontamination (OpenAI 2020) and remove training-evaluation overlaps based on problem names.

Training. We start our training process with DeepSeek-R1-Distill-Qwen-7B / 32B. In SFT, we train the models for 2 epochs with a learning rate of 2.0×10^{-5} and a batch size of 128 in the first stage, and 8 in the second stage. In RL, we use a batch size of 128, a learning rate of 1.0×10^{-6}, and train 450 steps for the 7B model and 350 steps for the 32B model. The

Model	FormalMATH-Lite		ProverBench		CombiBench	
	BEq@1	BEq@16	BEq@1	BEq@16	BEq@1	BEq@16
(*General-purpose Models*)						
OpenAI o3-pro	22.6	35.5	24.7	36.2	<u>9.0</u>	16.0
Claude4-thinking	20.8	32.2	24.4	35.6	**9.7**	<u>18.0</u>
Gemini-2.5-thinking	17.8	31.3	20.1	36.8	8.9	<u>18.0</u>
DeepSeek-R1-671B	18.4	31.3	23.5	34.5	8.1	**20.0**
DeepSeek-R1-Distill-7B	5.2	14.6	5.4	18.4	0.4	2.0
(*Specialized Models*)						
LeanFormalizer-PPO-7B	18.7	24.0	12.4	18.4	0.1	1.0
LeanFormalizer-SFT-7B	18.9	23.3	18.4	26.4	4.8	8.0
LeanFormalizer-CoT-7B	13.5	29.9	8.5	28.2	2.1	9.0
Herald-Translator-7B	13.6	24.7	8.2	27.0	1.3	5.0
Goedel-Formalizer-SonnetAnnotated-32B	18.7	29.2	13.6	27.6	3.4	10.0
Goedel-Formalizer-LeanWorkbookAnnotated-32B	15.1	26.4	5.0	12.1	0.3	2.0
Kimina-Autoformalizer-7B	35.1	<u>60.2</u>	13.3	25.3	2.6	6.0
(*Ours*)						
StepFun-Formalizer-7B	<u>38.3</u>	**61.2**	<u>25.1</u>	<u>37.9</u>	5.2	11.0
StepFun-Formalizer-32B	**40.5**	59.3	**26.7**	**38.5**	6.9	14.0

Table 3: BEq@1 and BEq@16 (%) results of StepFun-Formalizer and baselines on three benchmarks.

rollout temperature is 1.0. We use the Kimina Lean Server (Santos et al. 2025), equipped with 100 CPU cores and 400 GB of memory, under a 60-second time limit for equivalence verification. The SFT and RL stages are respectively executed on 8 and 32 A800-80G GPUs. The training of 7B and 32B take 45.38h and 55.85h. The context lengths are 16384. After training, we obtain StepFun-Formalizer-7B / 32B.

5.2 Experimental Settings

Benchmarks and Baselines To show the performance of StepFun-Formalizer, we evaluate it on both in-domain and out-of-distribution (OOD) benchmarks. For in-domain evaluation, we use FormalMATH-Lite (Yu et al. 2025b), which shares a similar distribution with our training data (Table 2). For OOD evaluation, we use ProverBench (Ren et al. 2025) and CombiBench (Liu et al. 2025a). Some formal statements in these two datasets include additional definitions, functions, and lemmas. We provide them as prompts and ask the model to generate main theorems, to evaluate its generalizability. The benchmarks encompass various areas, including algebra, number theory, and calculus, with difficulty levels ranging from high school to undergraduate.

We compare StepFun-Formalizer with advanced general-purpose LLMs, including o3-pro (OpenAI 2025), Claude4-thinking (Anthropic 2025), Gemini-2.5-thinking (Google 2025), DeepSeek-R1 and DeepSeek-R1-Distill-Qwen-7B (DeepSeek-AI 2025). Besides, we also evaluate specialized models, including LeanFormalizer (SJTULean 2024), Herald Translator (Gao et al. 2025), Goedel-Formalizer (Lin et al. 2025) and Kimina-Autoformalizer (Wang et al. 2025).

Metrics. We use the BEq@k (Liu et al. 2025b) metric to evaluate the models, which is the portion of samples where predicted statements are BEq to ground-truths at least once in k attempts:

$$\text{BEq@}k = \frac{1}{N}\sum_{i=1}^{N} \max_{j\in\{1,\cdots,k\}} \mathbb{1}(y_{i,j} \sim \hat{y}_i),$$

where N is the sample number; k is the attempt number; $\mathbb{1}(\cdot)$ is the indicator function, $\hat{y}_i$ is the ground-truth and $y_{i,j}$ is the j-th attempt for the i-th sample. We use BEq@1 and BEq@16 for evaluation. The temperature is set to 0.6.

5.3 Main Results

The evaluation results are shown in Table 3.

Our StepFun-Formalizer model establishes new state-of-the-art results on both FormalMATH-Lite (in-domain) and ProverBench (OOD), demonstrating the effectiveness and generalization of our data synthesis and training pipeline. Specifically, even StepFun-Formalizer-7B, surpasses every competing model on both benchmarks, offering a computational efficiency advantage: on FormalMATH-Lite it exceeds the previous best, Kimina-Autoformalizer-7B, which serves as its formal knowledge source, and on ProverBench it outperforms large general-purpose models such as DeepSeek-R1-671B. In addition, we observe that the 7B model performs comparably to or slightly better than the 32B model, which may be due to the limited size of the data. The 32B model may require more data for further improvement.

Our model outperforms specialized models with the same parameter scale in CombiBench. Since modelling combinatorial problems involves complex real-world scenarios and long contexts, it remains very challenging even for advanced general reasoning models to achieve high formalization accuracy on CombiBench, let alone smaller models. After training, our model shows a substantial improvement over specialized

models of the same scale, highlighting our model's capability in real-world scenario understanding.

5.4 Design Alternatives of Our Method

We explore design alternatives of our method, including the ablation study of the knowledge and reasoning datasets, another reasoning data collection method, and a different base model (see the extended version of the paper). To highlight how each choice impacts generalizable performance, we evaluate them on OOD benchmarks.

Both the knowledge and reasoning datasets contribute to the improvement of autoformalization. We conduct ablation studies to investigate the individual contributions of these two parts (knowledge and reasoning) of our data: StepFun-Formalizer-7B is re-trained with the same approach except removing the first stage (knowledge) and the second stage (reasoning) of SFT (Section 4.3) separately. The results (Table 4) show that informal-to-formal reasoning is the main contributor to model performance, while formal knowledge serves a complementary role. Notably, removing reasoning data leads to a sharp drop in BEq@16, highlighting its importance in boosting the performance upper bound.

Datasets	ProverBench		CombiBench	
	BEq@1	BEq@16	BEq@1	BEq@16
ThinkingF (Ours)	**25.1**	**37.9**	5.2	**11.0**
w/o Knowledge	24.5	37.4	3.9	10.0
w/o Reasoning	21.8	25.3	**5.3**	6.0

Table 4: Comparison of the roles of the two training datasets of the SFT stage.

The designed reasoning template helps the model better perform informal-to-formal translation. To show the effectiveness of the reasoning template, we replace the reasoning data in SFT with reasoning trajectories directly distilled from a general-purpose LLM. Specifically, we prompt Claude4-thinking (Anthropic 2025), a reasoning model of the same series as Claude 3.7 Sonnet, which we used to synthesize reasoning trajectories, to translate the problems in annotated datasets into formal statements, and use BEq to select the correct translations along with reasoning. The training results are shown in Table 5. It indicates that the reasoning process produced by Claude4-thinking causes a significant decline in the model's performance. We observe that, during formalization, the general reasoning model devotes its efforts to *solving* the informal problem instead of *formalizing* it, and thereby constraining its overall learning capability (see the extended version of the paper).

5.5 Further Analysis

Reinforcement learning can consistently improve the model's autoformalization capability. To show the performance improvement brought by reinforcement learning, we evaluate StepFun-Formalizer-7B every 50 training steps and record the average BEq@1 of all benchmarks. Both the reward and downstream task performance improve during training (see the extended version of the paper), which underscores the effectiveness of reinforcement learning.

Method	ProverBench		CombiBench	
	BEq@1	BEq@16	BEq@1	BEq@16
Template (Ours)	**25.1**	**37.9**	**5.2**	**11.0**
Direct Distillation	21.8	33.3	4.8	10.0

Table 5: Comparison between template-guided reasoning trajectory synthesis and direct distillation from a general-purpose reasoning model (Claude4-thinking).

Our model generates more verifiable formal statements in different domains. We conduct a simulation experiment to show the performance of StepFun-Formalizer in end-to-end theorem proving from natural language (Xuejun et al. 2025). Specifically, we randomly select 10K problems in NuminaMath-1.5, translate them into formal statements using StepFun-Formalizer-7B and Kimina-Autoformalizer-7B, and then use Kimina-Prover-7B (Wang et al. 2025) to generate 16 proofs for each statement. The results show that Kimina-Prover proves 4940 formal statements from StepFun-Formalizer-7B and 4549 from Kimina-Autoformalizer-7B. Detailed statistics of provable statements in each domain are shown in Table 6. Our model generates a higher proportion of provable formal statements across all domains except inequalities, highlighting its effectiveness in end-to-end theorem proving.

Domain	Kimina-7B	StepFun-Formalizer-7B (Ours)
Algebra	50.9	**55.6**
Number Theory	33.6	**36.1**
Logic and Puzzles	53.9	**58.4**
Inequalities	**20.3**	19.4
Calculus	29.5	**35.7**
Other	60.1	**64.3**
Total	45.5	**49.4**

Table 6: The proportion (%) of provable formal statements among 10K problems from NuminaMath-1.5 in each domain. "Kimina" refers to Kimina-Autoformalizer.

6 Conclusion

In this paper, we propose ThinkingF, a data synthesis and training pipeline for LLM-based autoformalization. The pipeline mitigates the lack of formal knowledge in general-purpose models via large-scale distillation and selection, and improves the model's understanding of natural language problems through template-guided reasoning synthesis, which facilitates complex informal-to-formal translation. We train 7B and 32B models with this pipeline and evaluate them on three benchmarks. Experimental results demonstrate that StepFun-Formalizer outperforms both general-purpose and specialized models on FormalMATH-Lite and ProverBench.

Acknowledgements

This work is partially supported by the Strategic Priority Research Program of the Chinese Academy of Sciences (Grants No.XDB0660300, XDB0660301, XDB0660302), Science and Technology Major Special Program of Jiangsu (Grant No. BG2024028), the NSF of China (Grants No. 62341411, 62222214, 6240073476), CAS Project for Young Scientists in Basic Research (YSBR-029) and Youth Innovation Promotion Association CAS.

References

Anthropic. 2025. Claude 4. https://www.anthropic.com/news/claude-4. Accessed: 2025-07-24.

Azerbayev, Z.; Piotrowski, B.; Schoelkopf, H.; Ayers, E. W.; Radev, D.; and Avigad, J. 2023. ProofNet: Autoformalizing and Formally Proving Undergraduate-Level Mathematics. arXiv:2302.12433.

Barras, B.; Boutin, S.; Cornes, C.; Courant, J.; Coscoy, Y.; Delahaye, D.; de Rauglaudre, D.; Filliâtre, J.-C.; Giménez, E.; Herbelin, H.; et al. 1999. The Coq proof assistant reference manual. *INRIA, version*, 6(11).

Beg, A.; O'Donoghue, D.; and Monahan, R. 2025. A Short Survey on Formalising Software Requirements using Large Language Models. arXiv:2506.11874.

Chen, J.; Tang, H.; Chu, Z.; Chen, Q.; Wang, Z.; Liu, M.; and Qin, B. 2024. Divide-and-Conquer Meets Consensus: Unleashing the Power of Functions in Code Generation. arXiv:2405.20092.

Chen, Y.; Liu, Y.; Zhou, J.; Hao, Y.; Wang, J.; Zhang, Y.; and Fan, C. 2025. R1-Code-Interpreter: Training LLMs to Reason with Code via Supervised and Reinforcement Learning. arXiv:2505.21668.

Community, L. 2023. Lean Documentation - lean3changes. https://lean-lang.org/lean4/doc/lean3changes.html. Accessed: 2025-07-23.

DeepSeek-AI. 2025. DeepSeek-R1: Incentivizing Reasoning Capability in LLMs via Reinforcement Learning. arXiv:2501.12948.

Gao, G.; Wang, Y.; Jiang, J.; Gao, Q.; Qin, Z.; Xu, T.; and Dong, B. 2025. Herald: A Natural Language Annotated Lean 4 Dataset. arXiv:2410.10878.

Google. 2025. Gemini 2.5: Pushing the Frontier with Advanced Reasoning, Multimodality, Long Context, and Next Generation Agentic Capabilities. arXiv:2507.06261.

Ji, X.; Liu, Y.; Wang, Q.; Zhang, J.; Yue, Y.; Shi, R.; Sun, C.; Zhang, F.; Zhou, G.; and Gai, K. 2025. Leanabell-Prover-V2: Verifier-integrated Reasoning for Formal Theorem Proving via Reinforcement Learning. arXiv:2507.08649.

Jiang, A. Q.; Li, W.; and Jamnik, M. 2023. Multilingual Mathematical Autoformalization. arXiv:2311.03755.

Kimi; Du, A.; Gao, B.; Xing, B.; Jiang, C.; Chen, C.; Li, C.; Xiao, C.; Du, C.; Liao, C.; et al. 2025. Kimi k1. 5: Scaling reinforcement learning with llms. *arXiv preprint arXiv:2501.12599*.

Li, Y.; Choi, D.; Chung, J.; Kushman, N.; Schrittwieser, J.; Leblond, R.; Eccles, T.; Keeling, J.; Gimeno, F.; Dal Lago, A.; Hubert, T.; Choy, P.; de Masson d'Autume, C.; Babuschkin, I.; Chen, X.; Huang, P.-S.; Welbl, J.; Gowal, S.; Cherepanov, A.; Molloy, J.; Mankowitz, D. J.; Sutherland Robson, E.; Kohli, P.; de Freitas, N.; Kavukcuoglu, K.; and Vinyals, O. 2022. Competition-level code generation with AlphaCode. *Science*, 378(6624): 1092–1097.

Li, Z.; Sun, J.; Murphy, L.; Su, Q.; Li, Z.; Zhang, X.; Yang, K.; and Si, X. 2024. A Survey on Deep Learning for Theorem Proving. arXiv:2404.09939.

Lin, X.; Cao, Q.; Huang, Y.; Wang, H.; Lu, J.; Liu, Z.; Song, L.; and Liang, X. 2024. FVEL: Interactive Formal Verification Environment with Large Language Models via Theorem Proving. arXiv:2406.14408.

Lin, Y.; Tang, S.; Lyu, B.; Wu, J.; Lin, H.; Yang, K.; Li, J.; Xia, M.; Chen, D.; Arora, S.; and Jin, C. 2025. Goedel-Prover: A Frontier Model for Open-Source Automated Theorem Proving. arXiv:2502.07640.

Liu, J.; Lin, X.; Bayer, J.; Dillies, Y.; Jiang, W.; Liang, X.; Soletskyi, R.; Wang, H.; Xie, Y.; Xiong, B.; Yang, Z.; Zhang, J.; Zhi, L.; Li, J.; and Liu, Z. 2025a. CombiBench: Benchmarking LLM Capability for Combinatorial Mathematics. arXiv:2505.03171.

Liu, Q.; Zheng, X.; Lu, X.; Cao, Q.; and Yan, J. 2025b. Rethinking and Improving Autoformalization: Towards a Faithful Metric and a Dependency Retrieval-based Approach. In *The Thirteenth International Conference on Learning Representations*.

Liu, Z.; Guo, X.; Lou, F.; Zeng, L.; Niu, J.; Wang, Z.; Xu, J.; Cai, W.; Yang, Z.; Zhao, X.; Li, C.; Xu, S.; Chen, D.; Chen, Y.; Bai, Z.; and Zhang, L. 2025c. Fin-R1: A Large Language Model for Financial Reasoning through Reinforcement Learning. arXiv:2503.16252.

Miranda, B.; Zhou, Z.; Nie, A.; Obbad, E.; Aniva, L.; Fronsdal, K.; Kirk, W.; Soylu, D.; Yu, A.; Li, Y.; and Koyejo, S. 2025. VeriBench: End-to-End Formal Verification Benchmark for AI Code Generation in Lean 4. In *2nd AI for Math Workshop @ ICML 2025*.

Moura, L. d.; and Ullrich, S. 2021. The Lean 4 Theorem Prover and Programming Language. In *Automated Deduction–CADE 28: 28th International Conference on Automated Deduction, Virtual Event, July 12–15, 2021, Proceedings 28*, 625–635. Springer.

OpenAI. 2020. Language Models are Few-Shot Learners. arXiv:2005.14165.

OpenAI. 2024a. GPT-4o System Card. arXiv:2410.21276.

OpenAI. 2024b. OpenAI o1 System Card. arXiv:2412.16720.

OpenAI. 2025. Introducing O3 and O4 Mini. https://openai.com/index/introducing-o3-and-o4-mini/. Accessed: 2025-07-24.

Paulson, L. C. 1994. *Isabelle: A Generic Theorem Prover*. Springer.

Qwen. 2025. Qwen3 Technical Report. arXiv:2505.09388.

Ren, Z. Z.; Shao, Z.; Song, J.; Xin, H.; Wang, H.; Zhao, W.; Zhang, L.; Fu, Z.; Zhu, Q.; Yang, D.; Wu, Z. F.; Gou,

Z.; Ma, S.; Tang, H.; Liu, Y.; Gao, W.; Guo, D.; and Ruan, C. 2025. DeepSeek-Prover-V2: Advancing Formal Mathematical Reasoning via Reinforcement Learning for Subgoal Decomposition. arXiv:2504.21801.

Renshaw, D. 2024. Compfiles: Catalog Of Math Problems Formalized In Lean. https://github.com/dwrensha/compfiles. Accessed: 2025-07-24.

Santos, M. D.; Wang, H.; de Saxcé, H.; Wang, R.; Baksys, M.; Unsal, M.; Liu, J.; Liu, Z.; and Li, J. 2025. Kimina Lean Server: Technical Report. arXiv:2504.21230.

Shao, Z.; Wang, P.; Zhu, Q.; Xu, R.; Song, J.; Bi, X.; Zhang, H.; Zhang, M.; Li, Y. K.; Wu, Y.; and Guo, D. 2024. DeepSeekMath: Pushing the Limits of Mathematical Reasoning in Open Language Models. arXiv:2402.03300.

SJTULean. 2024. LeanFormalizer_PPO. https://huggingface.co/SJTULean/LeanFormalizer_PPO. Accessed: 2025-07-30.

Thakur, A.; Lee, J.; Tsoukalas, G.; Sistla, M.; Zhao, M.; Zetzsche, S.; Durrett, G.; Yue, Y.; and Chaudhuri, S. 2025. CLEVER: A Curated Benchmark for Formally Verified Code Generation. arXiv:2505.13938.

Tsoukalas, G.; Lee, J.; Jennings, J.; Xin, J.; Ding, M.; Jennings, M.; Thakur, A.; and Chaudhuri, S. 2024. PutnamBench: Evaluating Neural Theorem-Provers on the Putnam Mathematical Competition. arXiv:2407.11214.

Wang, H.; Unsal, M.; Lin, X.; Baksys, M.; Liu, J.; Santos, M. D.; Sung, F.; Vinyes, M.; Ying, Z.; Zhu, Z.; Lu, J.; de Saxcé, H.; Bailey, B.; Song, C.; Xiao, C.; Zhang, D.; Zhang, E.; Pu, F.; Zhu, H.; Liu, J.; Bayer, J.; Michel, J.; Yu, L.; Dreyfus-Schmidt, L.; Tunstall, L.; Pagani, L.; Machado, M.; Bourigault, P.; Wang, R.; Polu, S.; Barroyer, T.; Li, W.-D.; Niu, Y.; Fleureau, Y.; Hu, Y.; Yu, Z.; Wang, Z.; Yang, Z.; Liu, Z.; and Li, J. 2025. Kimina-Prover Preview: Towards Large Formal Reasoning Models with Reinforcement Learning. arXiv:2504.11354.

Wang, X.; Wei, J.; Schuurmans, D.; Le, Q.; Chi, E.; Narang, S.; Chowdhery, A.; and Zhou, D. 2023. Self-Consistency Improves Chain of Thought Reasoning in Language Models. arXiv:2203.11171.

Weng, K.; Du, L.; Li, S.; Lu, W.; Sun, H.; Liu, H.; and Zhang, T. 2025. Autoformalization in the Era of Large Language Models: A Survey. arXiv:2505.23486.

Wu, Y.; Huang, D.; Shi, W.; Wang, W.; Pu, Y.; Gao, L.; Liu, S.; Nan, Z.; Yuan, K.; Zhang, R.; Zhang, X.; Du, Z.; Guo, Q.; Yin, D.; Hu, X.; and Chen, Y. 2025. InverseCoder: self-improving instruction-tuned code LLMs with inverse-instruct. In *Proceedings of the Thirty-Ninth AAAI Conference on Artificial Intelligence and Thirty-Seventh Conference on Innovative Applications of Artificial Intelligence and Fifteenth Symposium on Educational Advances in Artificial Intelligence*, AAAI'25/IAAI'25/EAAI'25. AAAI Press. ISBN 978-1-57735-897-8.

Wu, Y.; Jiang, A. Q.; Li, W.; Rabe, M. N.; Staats, C.; Jamnik, M.; and Szegedy, C. 2022. Autoformalization with Large Language Models. arXiv:2205.12615.

Xie, J.; Liu, C.; Yuan, Y.; Li, S.; Xiao, Z.; and Zhang, M. 2025. FMC: Formalization of Natural Language Mathematical Competition Problems. arXiv:2507.11275.

Xin, H.; Guo, D.; Shao, Z.; Ren, Z.; Zhu, Q.; Liu, B.; Ruan, C.; Li, W.; and Liang, X. 2024a. DeepSeek-Prover: Advancing Theorem Proving in LLMs through Large-Scale Synthetic Data.

Xin, H.; Ren, Z. Z.; Song, J.; Shao, Z.; Zhao, W.; Wang, H.; Liu, B.; Zhang, L.; Lu, X.; Du, Q.; Gao, W.; Zhu, Q.; Yang, D.; Gou, Z.; Wu, Z. F.; Luo, F.; and Ruan, C. 2024b. DeepSeek-Prover-V1.5: Harnessing Proof Assistant Feedback for Reinforcement Learning and Monte-Carlo Tree Search.

Xuejun, Y.; Zhong, J.; Feng, Z.; Zhai, P.; Yousefzadeh, R.; Ng, W. C.; Liu, H.; Shou, Z.; Xiong, J.; Zhou, Y.; Ong, C. B.; Sugiarto, A. J.; Zhang, Y.; Tai, W. M.; Cao, H.; Lu, D.; Sun, J.; Xu, Q.; Xin, S.; and Li, Z. 2025. Mathesis: Towards Formal Theorem Proving from Natural Languages. arXiv:2506.07047.

Yang, Z.; Chen, L.; Cohan, A.; and Zhao, Y. 2025. Table-R1: Inference-Time Scaling for Table Reasoning. arXiv:2505.23621.

Ying, H.; Wu, Z.; Geng, Y.; Yuan, Z.; Lin, D.; and Chen, K. 2025. Lean Workbook: A large-scale Lean problem set formalized from natural language math problems. arXiv:2406.03847.

Yu, Q.; Zhang, Z.; Zhu, R.; Yuan, Y.; Zuo, X.; Yue, Y.; Dai, W.; Fan, T.; Liu, G.; Liu, L.; Liu, X.; Lin, H.; Lin, Z.; Ma, B.; Sheng, G.; Tong, Y.; Zhang, C.; Zhang, M.; Zhang, W.; Zhu, H.; Zhu, J.; Chen, J.; Chen, J.; Wang, C.; Yu, H.; Song, Y.; Wei, X.; Zhou, H.; Liu, J.; Ma, W.-Y.; Zhang, Y.-Q.; Yan, L.; Qiao, M.; Wu, Y.; and Wang, M. 2025a. DAPO: An Open-Source LLM Reinforcement Learning System at Scale. arXiv:2503.14476.

Yu, Z.; Peng, R.; Ding, K.; Li, Y.; Peng, Z.; Liu, M.; Zhang, Y.; Yuan, Z.; Xin, H.; Huang, W.; Wen, Y.; Zhang, G.; and Liu, W. 2025b. FormalMATH: Benchmarking Formal Mathematical Reasoning of Large Language Models. arXiv:2505.02735.

Zhang, J.; Wang, Q.; Ji, X.; Liu, Y.; Yue, Y.; Zhang, F.; Zhang, D.; Zhou, G.; and Gai, K. 2025. Leanabell-Prover: Posttraining Scaling in Formal Reasoning. arXiv:2504.06122.

Zhang, L.; Valentino, M.; and Freitas, A. 2025. Formalizing Complex Mathematical Statements with LLMs: A Study on Mathematical Definitions. arXiv:2502.12065.

Zheng, K.; Han, J. M.; and Polu, S. 2022. MiniF2F: a cross-system benchmark for formal Olympiad-level mathematics. arXiv:2109.00110.

Zhou, J. P.; Staats, C.; Li, W.; Szegedy, C.; Weinberger, K. Q.; and Wu, Y. 2024. Don't Trust: Verify – Grounding LLM Quantitative Reasoning with Autoformalization. arXiv:2403.18120.

Zhu, Y.; Huang, D.; Lyu, H.; Zhang, X.; Li, C.; Shi, W.; Wu, Y.; Mu, J.; Wang, J.; Zhao, Y.; et al. 2025. CodeV-R1: Reasoning-Enhanced Verilog Generation. *arXiv preprint arXiv:2505.24183*.