Formalizing Complex Mathematical Statements with LLMs: A Study on Mathematical Definitions

Anonymous ACL submission

Abstract

Thanks to their linguistic capabilities, LLMs offer an opportunity to bridge the gap between informal mathematics and formal languages through *autoformalization*. However, it is still unclear how well LLMs generalize to sophisticated and naturally occurring mathematical statements. To address this gap, we investigate the task of autoformalizing real-world *mathematical definitions* – a critical component of mathematical discourse. Specifically, we introduce two novel resources for autoformalisation, collecting *definitions* from Wikipedia (Def_Wiki) and arXiv papers (Def_ArXiv). We then systematically evaluate a range of LLMs, analyzing their ability to formalize definitions into Isabelle/HOL. Furthermore, we investigate strategies to enhance LLMs' performance including *refinement through external feedback* from Proof Assistants, and *formal definition grounding*, where we guide LLMs through relevant contextual elements from formal mathematical libraries. Our findings reveal that definitions present a greater challenge compared to existing benchmarks, such as miniF2F. In particular, we found that LLMs still struggle with self-correction, and aligning with relevant mathematical libraries. At the same time, structured refinement methods and definition grounding strategies yield notable improvements of up to 16% on self-correction capabilities and 43% on the reduction of undefined errors, highlighting promising directions for enhancing LLM-based autoformalization in real-world scenarios.[1]

1 Introduction

Large Language Models (LLMs) have demonstrated remarkable potential in assisting with mathematical reasoning on different downstream tasks (Wei et al., 2022; Meadows et al., 2023, 2024; Valentino et al., 2022; Lu et al., 2023; Meadows and Freitas, 2023; Mishra et al., 2022a; Ferreira et al., 2022; Ferreira and Freitas, 2020; Welleck et al., 2021; Mishra et al., 2022b; Petersen et al., 2023). In the context of mathematics, formal languages play a crucial role by providing a precise, logic-based framework for verifying the correctness and logical validity of mathematical statements and proofs (Kaliszyk and Rabe, 2020). Consequently, one promising application of LLMs is *autoformalization*, the task of translating informal statements into formal languages (Wu et al., 2022). Given their advanced linguistic capabilities, LLMs offer an opportunity to bridge the gap between informal mathematics, natural language, and machine-verifiable logic, potentially streamlining and scaling the process of formal mathematical reasoning (Jiang et al., 2023; Tarrach et al., 2024).

[1]Code and datasets are available at `anonymized_link`

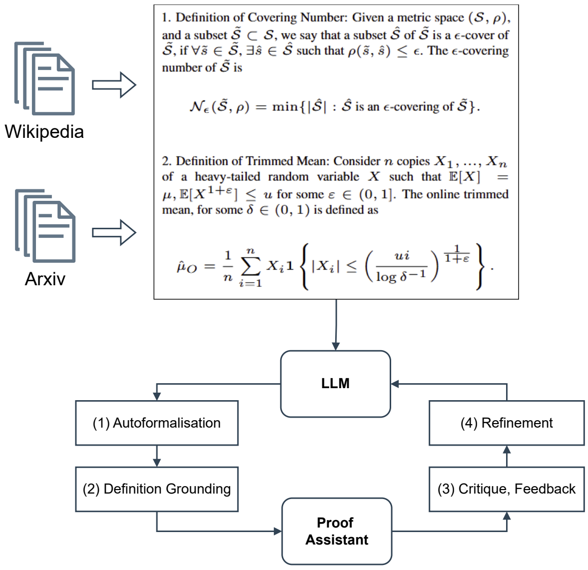


Figure 1: *Can LLMs formalize complex mathematical statements?* This paper investigates the task of translating *real-world mathematical definitions* into a formal language. We introduce a new resource collecting definitions from *Wikipedia* and *ArXiv* papers, exploring different strategies for autoformalization through the interaction between *LLMs* and *Proof Assistants*.

The task of autoformalization has garnered increasing attention in recent years, leading to the development of benchmarks and evaluation methodologies (Azerbayev et al., 2023; Zhang et al., 2024; Li et al., 2024). Despite this progress, however, existing benchmarks for autoformalization often focus on relatively simple mathematical problems, limiting our understanding of how well LLMs generalize to more sophisticated and naturally occurring mathematical statements.

To address this gap, this paper investigates the task of autoformalizing *mathematical definitions* – a critical component of mathematical discourse (Moschkovich, 2003). Definitions serve as foundational building blocks in mathematical reasoning, yet they are often intricate, context-dependent, and difficult to formalize. Evaluating LLMs on this subset of mathematical statements, therefore, allows for assessing their ability to formally represent fine-grained mathematical concepts, highlighting persisting challenges and limitations for real-world applications.

Specifically, this paper introduces two new benchmarks for autoformalization by collecting *real-world mathematical definitions* into two distinct resources: (1) *Def_Wiki*, including definitions extracted from Wikipedia articles, and (2) *Def_ArXiv*, including definitions collected from machine learning research papers. Using these resources, we first evaluate LLMs in a zero-shot setting, analyzing their ability to translate definitions into Isabelle/HOL (Nipkow et al., 2002).

Furthermore, to address observed limitations, we investigate two key strategies to enhance performance: (1) *Refinement via external feedback*, investigating the self-correction capabilities of LLMs by incorporating errors found by the supporting Proof Assistant. In particular, we show that while LLMs exhibit limited ability to refine outputs based on binary feedback (error vs. non-error), a more structured categorical refinement implemented via additional instructional constraints can improve performance. (2) *Formal definition grounding*. Many mathematical definitions require references to formal objects in external mathematical libraries. To guide LLMs in the process of autoformalization, we investigate the impact of introducing additional contextual control mechanisms, which add contextual elements from formal mathematical libraries as auxiliary premises.

Overall, our findings reveal that the proposed benchmarks present a greater challenge compared to existing autoformalization datasets, such as miniF2F (Zheng et al., 2022). In particular, LLMs struggle with self-correction and particularly with incorporating relevant mathematical libraries as preambles. Proposed structured refinement methods and definition grounding strategies both yield notable improvements, highlighting promising directions for enhancing LLM-based autoformalization in real-world scenarios.

Our contributions can be summarized as follows:

1. We introduce and release two novel datasets for autoformalization: Def_Wiki (definitions from Wikipedia) and Def_ArXiv (definitions from research papers on arXiv), designed to assess LLMs performance on complex, real-world mathematical definitions.

2. We perform a comprehensive error analysis on Isabelle/HOL, identifying key failures in formalizations generated by a range of LLMs, including GPT-4o (OpenAI et al., 2024a), Llama3 (Grattafiori et al., 2024) and DeepSeekMath (Shao et al., 2024).

3. We investigate refinement-based strategies, including structured feedback mechanisms from Proof Assistants and instruction-based categorical refinements.

4. We explore the role of formal definition grounding, investigating how the inclusion of relevant mathematical libraries impacts the ability of LLMs to connect the formalized statements with contextual mathematical elements and relevant premises. We found that definition grounding is fundamental for complex autoformalization.

2 Autoformalization with LLMs

The task of autoformalization can be defined as a transformation function from natural language and LaTeX symbols $\mathcal{S}$ to a formal language $\mathcal{F}$, $f : \mathcal{S} \to \mathcal{F}$, such that for every informal mathematical statement $s \in \mathcal{S}$, there exists a formal mathematical statement $\phi \in \mathcal{F}$ where $f(s) = \phi$ (Zhang et al., 2024). Autoformalization via LLMs reifies the transformation function as:

$$f(s) = \text{LLM}(p_{\text{auto}}, \{(s_i, \phi_i)\}, s),$$

where p_{auto} is a prompt for autoformalization and $\{(s_i, \phi_i)\}$ is an optional set of exemplars.

miniF2F	Def_Wiki	Def_ArXiv
1. Suppose that $\sec x + \tan x = \frac{22}{7}$ and that $\csc x + \cot x = \frac{m}{n}$, where $\frac{m}{n}$ is in lowest terms. Find $m+n$. Show that it is 044. 2. What is the sum of the two values of x for which $(x+3)^2 = 121$? Show that it is -6. 3. The product of two positive whole numbers is 2005. If neither number is 1, what is the sum of the two numbers? Show that it is 406. 4. The expression $10x^2 - x - 24$ can be written as $(Ax-8)(Bx+3)$, where A and B are integers. What is $AB+B$? Show that it is 12.	1. Definition of Rademacher Complexity: Given a set $A \subseteq \mathbb{R}^m$, the Rademacher complexity of A is defined as follows: $\mathrm{Rad}(A) := \frac{1}{m}\mathbb{E}_\sigma \left[\sup_{a \in A} \sum_{i=1}^{m} \sigma_i a_i\right]$ where $\sigma_1, \sigma_2, \ldots, \sigma_m$ are independent random variables drawn from the Rademacher distribution (i.e. $\Pr(\sigma_i = +1) = \Pr(\sigma_i = -1) = 1/2$ for $i = 1, 2, \ldots, m$), and $a = (a_1, \ldots, a_m)$. 2. Definition of Polynomial Kernel: For degree-d polynomials, the polynomial kernel is defined as $K(\mathbf{x}, \mathbf{y}) = (\mathbf{x}^\mathsf{T}\mathbf{y} + c)^d$ where $\mathbf{x}$ and $\mathbf{y}$ are vectors of size n in the input space, i.e. vectors of features computed from training or test samples and $c \geq 0$ is a free parameter trading off the influence of higher-order versus lower-order terms in the polynomial.	1. Definition of Covering Number: Given a metric space $(\mathcal{S}, \rho)$, and a subset $\tilde{\mathcal{S}} \subset \mathcal{S}$, we say that a subset $\hat{\mathcal{S}}$ of $\tilde{\mathcal{S}}$ is a ϵ-cover of $\tilde{\mathcal{S}}$, if $\forall \tilde{s} \in \tilde{\mathcal{S}}, \exists \hat{s} \in \hat{\mathcal{S}}$ such that $\rho(\tilde{s}, \hat{s}) \leq \epsilon$. The ϵ-covering number of $\tilde{\mathcal{S}}$ is $\mathcal{N}_\epsilon(\tilde{\mathcal{S}}, \rho) = \min\{\lvert\hat{\mathcal{S}}\rvert : \hat{\mathcal{S}} \text{ is an } \epsilon\text{-covering of } \tilde{\mathcal{S}}\}.$ 2. Definition of Trimmed Mean: Consider n copies $X_1, \ldots, X_n$ of a heavy-tailed random variable X such that $\mathbb{E}[X] = \mu, \mathbb{E}[X^{1+\varepsilon}] \leq u$ for some $\varepsilon \in (0, 1]$. The online trimmed mean, for some $\delta \in (0, 1)$ is defined as $\hat{\mu}_O = \frac{1}{n}\sum_{i=1}^{n} X_i \mathbf{1}\left\{\lvert X_i\rvert \leq \left(\frac{ui}{\log \delta^{-1}}\right)^{\frac{1}{1+\varepsilon}}\right\}.$

Table 1: Examples of instances from Def_Wiki and Def_ArXiv and comparison with miniF2F.

Property	miniF2F-Test	Def_Wiki	Def_ArXiv
No. Samples	244	56	30
No. Tokens	70.25(47.70)	200.18(112.98)	164.40(71.47)
No. Objects	4.76(1.68)	7.63(2.71)	7.10(2.64)
No. Formulae	2.71(1.74)	2.84(2.05)	3.17(1.97)

Table 2: Dataset properties. The number of tokens per sample is calculated using the GPT-2 tokenizer. The number of directly mentioned mathematical objects—excluding explicit numbers and variables—and the number of mathematical formulae per sample are estimated through prompting with GPT-4o. The mean (standard deviation) is reported for each dataset.

2.1 Limitations of Existing Benchmarks

Naturally occurring mathematical statements typically involve complex and abstract mathematical concepts. However, the statements in existing datasets, such as miniF2F (Zheng et al., 2022), primarily consist of basic arithmetic operations and elementary mathematical objects, such as integers, fractions, and real numbers (as shown in Table 1). Such mathematical objects are relatively simple compared to the complex and abstract concepts found in naturally occurring mathematical statements and scientific papers, which may involve higher-level structures like vectors, matrices, and probability. The operations are also limited to simple arithmetic, such as addition, subtraction, multiplication, division, and exponentiation. Studying autoformalization on such datasets, therefore, does not necessarily reflect the challenges of autoformalization in realistic scenarios.

In addition, the ground-truth formal code in publicly available datasets may have been exposed to LLMs whose training corpora are not disclosed. Fundamental mathematical definitions also have a high likelihood of already being formalized in theorem prover libraries. This risk raises concerns about data leakage when LLMs perform autoformalization and could lead to biased results when analyzing performance. However, few benchmarks focus on complex mathematical definitions. To address this, we propose constructing data samples of mathematical definitions in the machine learning domain, as concepts in this area are sufficiently complex and less likely to have been formalized.

2.2 Mathematical Definitions in Machine Learning Domain

We obtain mathematical definitions in machine learning domain from two sources: Wikipedia (Def_Wiki) and Arxiv Papers (Def_ArXiv). Definitions from these two sources are likely to have already been validated and exhibit sufficient variety. For Def_Wiki, definitions are from pages under the Machine Learning category[2] and its sub-categories. We manually browsed each page, identified well-defined definitions (i.e., formal descriptions with mathematical symbols), and converted the chosen definitions into LaTeX format. In total, we obtained 56 qualified natural language definitions in LaTeX and divided them into development and test sets, containing 10 and 46 samples, respectively. For Def_ArXiv, we used the advanced search tool on ArXiv's website, filtering for papers in the subcategories cs.LG and stat.ML, with comments including "ICML." We restricted the search to papers published in 2019, 2020, and 2021 and manually reviewed the first 25 papers from each year. We shortlisted papers that were accepted to the ICML conference and contained formally described definitions with mathematical symbols to ensure reli-

[2] https://en.wikipedia.org/wiki/Category:Machine_learning

ability. We then filtered out definitions that were less straightforward or formal in their expressions, extracted the LaTeX versions, and ultimately obtained 30 definitions from 7 papers.

Statements in definition datasets are more abstract and complex, as intuitively shown in the randomly chosen examples in Table 1. The data properties are summarized in Table 2. Although small in scale, definition datasets exhibit higher means for the number of tokens, mathematical objects, and formulae per example, indicating that they are more challenging. Additionally, definition datasets have higher standard deviations, suggesting greater diversity among the samples.

The data samples in our benchmarks contain only definitions in LaTeX format. We did not include ground-truth formal codes for the following reasons: 1. Including ground-truth formal codes could increase the risk of the aforementioned data leakage problem. 2. A single mathematical statement can have multiple correct formalizations. An autoformalized code that differs from the ground-truth does not necessarily indicate incorrect formalization. 3. The purpose of ground-truth formal codes is to evaluate autoformalization. However, the syntactic correctness of formalized code can be rigorously and automatically verified using theorem provers (Zhang et al., 2024), and semantic consistency can be evaluated in a reference-free manner (Li et al., 2024). Manual inspection of autoformalized code also does not require ground-truth formal codes.

3 Empirical Evaluation

Empirical Setup. Isabelle/HOL is chosen as the representative formal language due to its wide adoption and support for formal mathematical reasoning. We evaluate three LLMs with different features: DeepSeekMath-7B (Shao et al., 2024), Llama3-8B (Grattafiori et al., 2024) and GPT-4o (OpenAI et al., 2024a). DeepSeekMath-7B is an open-sourced LLM trained specifically for mathematics using mathematical contents from Common Crawl. As a smaller model, it has demonstrated comparable mathematical reasoning performance as in GPT-4 (OpenAI et al., 2024b), and strong few-shot autoformalization performance on miniF2F with Isabelle. This superiority makes it a good representative of smaller but specialized LLMs. LLama3-8B is a smaller open-sourced foundation LLM with no specific emphasis on math. GPT-4o is widely acknowledged as one of the state-of-the-art LLMs. For reproducibility, greedy decoding is used for generation in all settings.

Evaluation Metrics. The success rate of passing the check by the Isabelle Proof Assistant across the tested dataset is used as the first metric. We assume that a formalized code instance with the first error occurring later in the code reflects a more complete autoformalization. Thus, we evaluate such by calculating the proportion of correct lines (up to the first error) within the main body of the code. For syntactically correct instances, this value is equal to 1. To better monitor the occurrence of errors, we group them into three categories: Syntax Errors (SYN), Undefined Item Errors (UDF), and Type Unification Failed Errors (TUF). For each category, we calculate the percentage of incorrect formalized codes caused by errors in that category.

3.1 Zero-Shot Prompting & Binary Refinement

In order to understand the challenges in autoformalising mathematical definitions with LLMs, we perform a preliminary analysis on miniF2F (Zheng et al., 2022), Def_Wiki and Def_ArXiv using zero-shot prompting (ZS) and binary refinement. With binary refinement, we aim to assess the capabilities of LLMs for error correction, providing them with the formal code generated via ZS, along with the syntactic correctness evaluated using the proof assistant (i.e., “correct”, “incorrect”). From the results reported in Table 3, we can derive the following observations:

Def_Wiki and Def_ArXiv are more challenging than miniF2F. When performing autoformalization on Def_Wiki and Def_ArXiv, GPT-4o achieves a significantly lower success rates (-13.78% on average) and FEO (-31.90% on average) compared to results on miniF2F-Test.

LLMs can provide false preambles when performing autoformalization. In Table 3, we observe that the percentage of Invalid Inputs errors (IVI) can be non-zero. Errors in this category are caused by either non-existent preambles or invalid theory file formats in structure. For Llama3-8B the latter is more common whereas for GPT-4o, we observe that the dominant cause is the generation of non-existent preambles. This behaviour shows that LLMs do not perfectly generalize in recognizing the names of preambles.

Prompt Strategy	Model	Pass↑	FEO↑	TRO↓	IVI↓	SYN↓	UDF↓	TUF↓
miniF2F-Test								
ZS	DeepSeekMath-7B	3.28	12.79	18.44	**0.00**	50.00	14.34	9.43
ZS + Binary		2.05	6.73	2.46	**0.00**	79.91	**5.33**	**2.05**
ZS	Llama3-8B	4.92	20.70	4.51	0.41	29.51	38.52	18.85
ZS + Binary		3.69	20.52	3.28	0.41	33.20	39.75	20.49
ZS	GPT-4o	25.41	48.90	**1.23**	1.23	**6.15**	23.77	7.38
ZS + Binary		**29.10**	**53.90**	2.05	1.23	**6.15**	21.72	8.20
Def_Wiki-Test								
ZS	DeepSeekMath-7B	10.87	17.75	34.78	2.17	30.43	26.09	**2.17**
ZS + Binary		6.52	7.73	8.70	**0.00**	69.57	**21.74**	**2.17**
ZS	Llama3-8B	0.00	2.80	**0.00**	23.91	56.52	32.61	4.35
ZS + Binary		2.17	3.71	**0.00**	26.09	52.17	30.43	**2.17**
ZS	GPT-4o	10.87	16.12	8.70	8.70	19.57	50.00	13.04
ZS + Binary		**13.04**	**18.30**	8.70	6.52	**17.39**	50.00	13.04
Def_ArXiv								
ZS	DeepSeekMath-7B	13.33	14.69	16.67	**0.00**	40.00	36.67	13.33
ZS + Binary		3.33	3.33	6.67	**0.00**	66.67	**33.33**	**3.33**
ZS	Llama3-8B	0.00	2.67	**0.00**	13.33	70.00	40.00	6.67
ZS + Binary		3.33	5.83	**0.00**	20.00	60.00	**33.33**	6.67
ZS	GPT-4o	13.33	19.30	**0.00**	**0.00**	40.00	56.66	6.67
ZS + Binary	GPT-4o	**16.67**	**24.30**	**0.00**	**0.00**	**33.33**	53.33	6.67

Table 3: Autoformalization results. Prompt strategies include: (**ZS**): zero-shot autoformalization; (**ZS + Binary**): refinement given (zero-shot) formalized code and binary syntactic correctness state. Pass rate (**Pass**), the place of first error occurrence in the main body of the code (**FEO**), and percentage of occurrence of each error category are recorded here. Errors in each error category are: (**TRO**): Time Run-Out for checking; (**IVI**): Fake Non-Existent Theory, Invalid structural format; (**SYN**): Inner syntax error, Outer syntax error, Inner lexical error, Malformed command syntax, Bad name, Bad number of arguments for type constructor, Extra free type variable(s); (**UDF**): Undefined type names, Undeclared class, Undefined locale, No type arity list, Extra variables on rhs; (**TUF**) Type unification failed.

Specialized smaller models can reach the same level of success rate as larger LLMs. As a model designed specifically for mathematics, DeepSeekMath with 7B parameters can achieve a similar success rate as GPT-4o. Although Llama3-8B has a larger model size, its generalization ability on definitions is limited. Additionally, DeepSeekMath-7B exhibits a lower percentage of undefined type names errors (UDF). However, one disadvantage of the specialized model is that its formalizations have a higher percentage of time run-out issues (TRO). This is likely caused by the bias introduced during the fine-tuning phase on theorem proving which can lead the model to generate unsolicited proofs.

Small LLMs possess limited binary self-correction capabilities. With binary refinement, GPT-4o produces formal codes with a higher success rate on all three datasets, whereas for DeepSeekMath-7B this mechanism leads to a performance decrease. LLama3-8B also fails to self-correct its autoformalization results on miniF2F. This behavior suggests that self-refinement exceeds the capabilities of smaller LLMs.

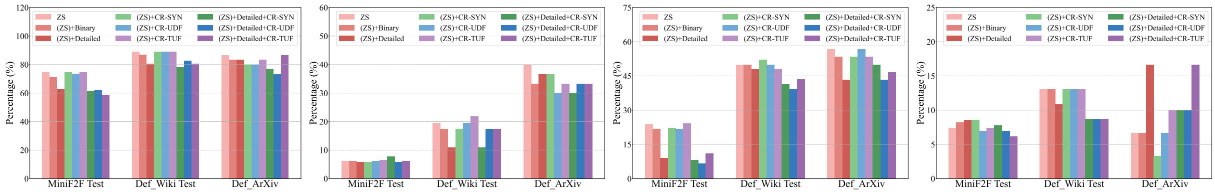
3.1.1 Error Analysis & Interventions

To understand potential interventions for improving autoformalization, we qualitatively analyze error patterns on the development set of Def_Wiki. Our analysis is based on the results obtained via GPT-4o, given its better performances on ZS and binary refinement. The main reasons for failure identified through our analysis are summarized in Table 5, with additional examples reported in Appendix.

We observe that syntactic errors (SYN) exhibit the most variety, suggesting that GPT-4o may struggle to follow syntactic rules in Isabelle/HOL if not explicitly instructed. Type unification errors (TUF) suggest that GPT-4o my struggle with the exact usage of defined Isabelle items. To improve these issues, we investigate a **Categorical Refinement** (CR) method. CR involves designing specific additive instructions that constraint the behaviors leading to errors identified in the qualitative analysis.

Similarly, for syntactic errors (SYN), causes 1, 2, and 3 in Table 4 can be addressed with rule-based algorithms that refine formal codes at the symbolic level (**Symbolic Refinement**, SR).

Undefined errors (UDF), on the other hand, indicate that although GPT-4o can refer to external formal mathematical items, it remains unaware of the location or existence of relevant libraries. To alleviate UDF errors, we propose the process of **Formal Definition Grounding** (FDG) based on two methods: 1. *Postprocessing* (Post-FDG): explicitly augment preambles generated by LLMs with



(a) Overall Error Rate (b) SYN Error Rate (c) UDF Error Rate (d) TUF Error Rate

Figure 2: Error rates of different refinement methods on GPT-4o. Variants include: (**ZS**): zero-shot autoformalization; (**(ZS)+Binary**): binary refinement on (zero-shot) formal codes; (**(ZS)+Detailed**): detailed refinement on (zero-shot) formal codes; (**(ZS)+CR-SYN/UDF/TUF**): plain refinement on (zero-shot) formal codes with SYN/UDF/TUF categorical refinement instructions; (**(ZS)+Detailed+CR-SYN/UDF/TUF**): detailed refinement on (zero-shot) formal codes with SYN/UDF/TUF categorical refinement instructions.

relevant libraries; 2. *Prompting* (Prompt-FDG): provide LLMs with grounded formal items and preambles in context to guide autoformalization.

3.2 Categorical Refinement

In order to better understand the refinement capabilities of GPT4-o, we investigate a set of error correction strategies: (i) Plain: provide LLMs with previously generated formal codes; (ii) Binary: additionally, provide LLMs with the correctness status of the formal code; (iii) Detailed: instead of just the binary correctness status, provide LLMs with the details of type, message, and line location of individual errors in the code.

In addition, to evaluate categorical refinement, we design specific instructions for each category of errors based on our qualitative analysis (Table 5). We report the error rate results of different refinement methods on GPT-4o in bar charts in Figure 2. All prompts used for categorical refinement along with additional empirical results are provided in Appendix.

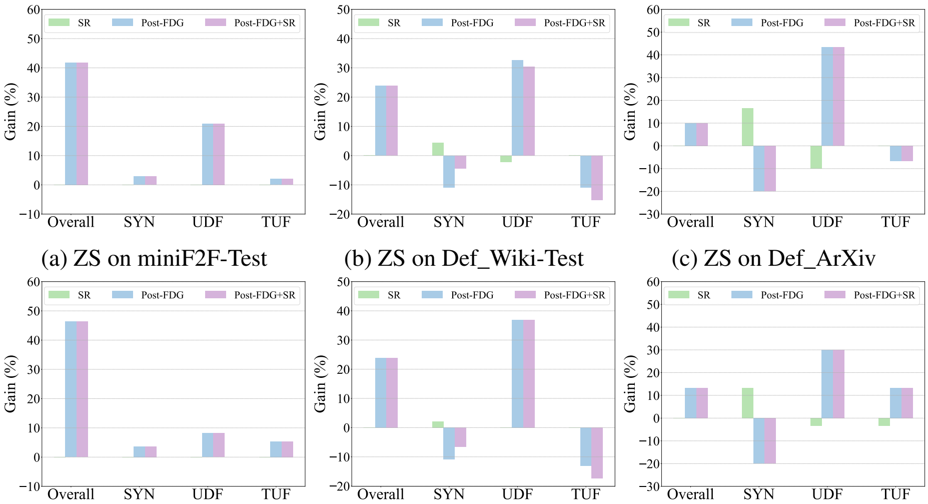
Providing LLMs with more information about individual errors is more effective than simply indicating binary correctness. As shown in Figure 2a, both binary and detailed refinements can reduce the overall error rate across all the datasets, with detailed refinement fixing more errors on miniF2F-Test and Def_Wiki-Test. For SYN errors, although there is no clear trend indicating that one refinement outperforms the other, both refinements lead to a lower error rate compared to zero-shot autoformalization. Detailed refinement also decreases the percentage of UDF errors as shown in Figure 2c. These performance gains suggest that detailed refinement improves the quality of autoformalized codes. For TUF errors, applying both refinements does not consistently result in a lower error rate, indicating that errors in this category are more difficult for LLMs to fix.

Categorical refinement demonstrates superiority in reducing error rates. As shown in Figure 2a, across all datasets, the refinement method that achieves the lowest overall error rate incorporates one of the instructions for categorical refinement, highlighting its superiority. However, when categorical refinement is applied without error details, such improvements do not occur. We hypothesize that this is because categorical instructions serve as constraints, making it more difficult for the LLM to follow them without more detailed error information for individual instances. Once such information is provided, the LLM receives sufficient information to adhere to the categorical refinement instructions.

Categorical refinement can effectively reduce errors for specific categories. As shown in Figure 2b, the method with the lowest SYN error rate on miniF2F-Test is plain refinement with SYN categorical refinement instructions, whereas on the other two datasets the best performing method is SYN categorical refinement with error details. In Figure 2c, UDF categorical refinement with error details also leads to the lowest UDF error rate on all three datasets. Similarly in Figure 2d, TUF categorical refinement with error details achieves the lowest TUF error rate on two out of the three datasets. These results collectively demonstrate the effectiveness of the design of categorical refinement. The only exceptional is TUF errors on the Def_ArXiv dataset, which again highlights the difficulty of fixing TUF errors.

3.3 Symbolic Refinement

Based on reasons 1 and 2 of SYN errors in Table 5, we defined two rules for implementing Symbolic



(a) ZS on miniF2F-Test (b) ZS on Def_Wiki-Test (c) ZS on Def_ArXiv

(d) ZS+Detailed on miniF2F (e) ZS+Detailed on Def_Wiki (f) ZS+Detailed on Def_ArXiv

Figure 3: Gain of error rates when testing autoformalization with different methods compared to direct test. We evaluate results on zero-shot autoformalized codes and (zero-shot) formal codes with detailed refinement. Testing variants include: (**SR**): Symbolic Refinement; (**Post-FDG**): Postprocessing with Formal Definition Grounding.

Refinement: (1) if a symbol in the formal code is likely to be an Isabelle symbol (i.e., it starts with "\<" but misses ">"), we add ">" at its end to ensure that the symbol follows Isabelle's format; (2) for non-existent symbols of mathematical fonts, we replace them with relevant symbols in Isabelle.

The differences in error rates between our methods and direct testing results are shown as bar charts in Figure 3. Detailed numbers and additional results are provided in *Appendix*.

Symbolic Refinement can effectively reduce SYN errors in the generated formal codes on definition datasets. In Figures 3b and 3e, both applying SR alone and in combination with Post-FDG lead to a lower SYN error rate on Def_Wiki-Test. On Def_ArXiv, Figures 3c and 3f similarly shows that applying SR alone results in a reduction of SYN errors. These results suggest that SR is an effective approach for addressing SYN errors. On miniF2F-Test, however, SR does not influence the error rates. This is because SR is closely tied to specific error patterns in the dataset.

3.4 Formal Definition Grounding (FDG)

3.4.1 FDG via Post-Processing

For FDG, we first extracted external formal definitions of mathematical items and their sources from the Isabelle/HOL library. Then we filtered the extracted definitions to retain only those likely relevant to the autoformalization task on the datasets. Finally, for each individual instance in Def_Wiki and Def_ArXiv, we manually determined which formal definitions should be provided as contextual elements. For miniF2F, we simply selected the definitions of real and complex numbers as the relevant definitions.

Autoformalization performance can be underestimated without including contextual information. In Figure 3, without modifying the main body of the formalization, replacing the preambles with possible preambles via FDG (Post-FDG) directly leads to higher overall syntactic correctness. On miniF2F-Test, this setting only considers sources containing formal definitions of real and complex numbers, yet it increases overall syntactic correctness by more than 40%.

FDG can reduce the occurrence of errors caused by referring to undefined mathematical objects. In Figure 3, the UDF error category has the most significant improvement from Post-FDG. Even when LLMs do not include the exact library that contains relevant mathematical items, they tend to use conventional names for the autoformalization task. By importing the appropriate theory files, these previously undefined items can be linked to the formalization, thereby reducing UDF errors.

Errors in autoformalized codes for definition datasets are more likely to be entangled than those in the miniF2F dataset. In Figure 3a and Figure 3d, Post-FDG leads to positive performance gains across all error categories. However, in Figures 3b, 3c, 3e and 3f, while UDF error rates de-

Prompt Strategy	Pass↑	SYN↓	UDF↓	TUF↓
ZS	34.78	30.43	17.39	23.91
Soft-IFDC	19.57	34.78	30.43	26.09
Hard-IFDC	19.57	36.96	21.74	39.13

Table 4: GPT-4o Error results of Prompt-FDG on Def_Wiki-Test with Post-FDG applied. **IFDC**: provide LLM with formal definition codes from FDG and force (**Hard**) or not force (**Soft**) LLM to use them.

crease, error rates in other categories can increase. A similar trend is observed when applying SR, where a reduction in SYN errors can coincide with increases in errors from the other two categories. This phenomenon suggests that because definition datasets are more complex, LLMs are more prone to generating entangled errors during the autoformalization process.

3.4.2 FDG via Prompting

We designed two prompts to include external formal definitions for FDG: 1. Soft: allow the LLM some flexibility in whether to use in-context formal definitions for autoformalization; 2. Hard: explicitly instruct the LLM to use the in-context formal definitions if they are related. We tested these prompts on GPT-4o and Def_Wiki-Test to evaluate whether it can correctly refer to formalised items in context. The results are reported in Table 4.

Including relevant formal definitions in the prompt does not boost the performance of autoformalization. Intuitively, LLMs should perform better when more relevant information is provided within the prompt. However, directly including grounded formal definitions does not positively impact the formalisation. This behaviour indicates that current state-of-the-art LLMs cannot effectively link to relevant in-context formal items for autoformalization.

4 Related Work

Autoformalization builds connections between natural language and formal languages (Quan et al., 2024b,a). It also plays an important role in formal mathematical reasoning. For instance, proof autoformalization has been used as an intermediate step in automated theorem proving (Jiang et al., 2023; Tarrach et al., 2024). Deep learning models, such as transformers, have been applied to autoformalization. For example, Cunningham et al. (2022) developed a transformer-based model for autoformalizing of theorems along with their proofs in Coq. In recent years, with the increasing capabilities of LLMs, prompting-based methods have also demonstrated the ability to autoformalize mathematical statements in Isabelle (Wu et al., 2022; Zhang et al., 2024) and Lean (Li et al., 2024). Despite recent progress in autoformalization, few studies have analyzed this task from an error perspective. Our work aim to address this gap.

There are a few benchmarks providing informal-formal mathemtical statement pairs. miniF2F dataset (Zheng et al., 2022) provided 488 mathematical statement pairs from high school and undergraduate level to Olympiad problems in Lean, Metamath, and Isabelle. ProofNet (Azerbayev et al., 2023) benchmark contains 371 parallel formal theorem statements, natural language theorem statements, and natural language proofs in Lean. However, informal-Formal mathematical statement pairs are still scarce. Obtaining ground-truth formal codes requires specialists and there are many ways to formalize mathematical statements. In our work, we only provide the datasets in the natural language version and aim to develop methods without ground-truth formal codes.

5 Conclusion

This paper explored the challenges and advancements in autoformalization of complex mathematical statements. To this end, two datasets collecting real-world definitions in machine learning were introduced for systematic evaluation. By assessing autoformalization performance across three modern LLMs on newly introduced datasets, we identify key failure patterns including syntactic inconsistency, undefined references, and type mismatch. To address these, we proposed interventions such as Formal Definition Grounding and Categorical Refinement to enhance performance. Our results suggest that while modern LLMs exhibit potential in converting natural language mathematical definitions into formal representations, they still require improved guidance mechanisms and structured refinement techniques to enhance accuracy. Future research should focus on strengthening self-correction capabilities and integrating more robust contextual understanding into LLM-based formalization systems.

6 Limitations

Despite its contributions, this study has several limitations. First, the error analysis was conducted in Isabelle/HOL, and results may not directly generalize to other formal proof assistants such as Lean. Second, the definition datasets proposed, though diverse, are relatively small scale. Additionally, while the proposed refinements improve formalization performance, they do not fully eliminate semantic inconsistencies between natural language definitions and their formalized counterparts. More advanced methods are still needed to be developed.

References

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

Garett Cunningham, Razvan Bunescu, and David Juedes. 2022. Towards autoformalization of mathematics and code correctness: Experiments with elementary proofs. In *Proceedings of the 1st Workshop on Mathematical Natural Language Processing (MathNLP)*, pages 25–32, Abu Dhabi, United Arab Emirates (Hybrid). Association for Computational Linguistics.

Deborah Ferreira and André Freitas. 2020. Premise selection in natural language mathematical texts. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7365–7374.

Deborah Ferreira, Mokanarangan Thayaparan, Marco Valentino, Julia Rozanova, and Andre Freitas. 2022. To be or not to be an integer? encoding variables for mathematical text. In *Findings of the Association for Computational Linguistics: ACL 2022*, pages 938–948, Dublin, Ireland. Association for Computational Linguistics.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, Amy Yang, Angela Fan, Anirudh Goyal, Anthony Hartshorn, Aobo Yang, Archi Mitra, Archie Sravankumar, Artem Korenev, Arthur Hinsvark, Arun Rao, Aston Zhang, Aurelien Rodriguez, Austen Gregerson, Ava Spataru, Baptiste Roziere, Bethany Biron, Binh Tang, Bobbie Chern, Charlotte Caucheteux, Chaya Nayak, Chloe Bi, Chris Marra, Chris McConnell, Christian Keller, Christophe Touret, Chunyang Wu, Corinne Wong, Cristian Canton Ferrer, Cyrus Nikolaidis, Damien Allonsius, Daniel Song, Danielle Pintz, Danny Livshits, Danny Wyatt, David Esiobu, Dhruv Choudhary, Dhruv Mahajan, Diego Garcia-Olano, Diego Perino, Dieuwke Hupkes, Egor Lakomkin, Ehab AlBadawy, Elina Lobanova, Emily Dinan, Eric Michael Smith, Filip Radenovic, Francisco Guzmán, Frank Zhang, Gabriel Synnaeve, Gabrielle Lee, Georgia Lewis Anderson, Govind Thattai, Graeme Nail, Gregoire Mialon, Guan Pang, Guillem Cucurell, Hailey Nguyen, Hannah Korevaar, Hu Xu, Hugo Touvron, Iliyan Zarov, Imanol Arrieta Ibarra, Isabel Kloumann, Ishan Misra, Ivan Evtimov, Jack Zhang, Jade Copet, Jaewon Lee, Jan Geffert, Jana Vranes, Jason Park, Jay Mahadeokar, Jeet Shah, Jelmer van der Linde, Jennifer Billock, Jenny Hong, Jenya Lee, Jeremy Fu, Jianfeng Chi, Jianyu Huang, Jiawen Liu, Jie Wang, Jiecao Yu, Joanna Bitton, Joe Spisak, Jongsoo Park, Joseph Rocca, Joshua Johnstun, Joshua Saxe, Junteng Jia, Kalyan Vasuden Alwala, Karthik Prasad, Kartikeya Upasani, Kate Plawiak, Ke Li, Kenneth Heafield, Kevin Stone, Khalid El-Arini, Krithika Iyer, Kshitiz Malik, Kuenley Chiu, Kunal Bhalla, Kushal Lakhotia, Lauren Rantala-Yeary, Laurens van der Maaten, Lawrence Chen, Liang Tan, Liz Jenkins, Louis Martin, Lovish Madaan, Lubo Malo, Lukas Blecher, Lukas Landzaat, Luke de Oliveira, Madeline Muzzi, Mahesh Pasupuleti, Mannat Singh, Manohar Paluri, Marcin Kardas, Maria Tsimpoukelli, Mathew Oldham, Mathieu Rita, Maya Pavlova, Melanie Kambadur, Mike Lewis, Min Si, Mitesh Kumar Singh, Mona Hassan, Naman Goyal, Narjes Torabi, Nikolay Bashlykov, Nikolay Bogoychev, Niladri Chatterji, Ning Zhang, Olivier Duchenne, Onur Çelebi, Patrick Alrassy, Pengchuan Zhang, Pengwei Li, Petar Vasic, Peter Weng, Prajjwal Bhargava, Pratik Dubal, Praveen Krishnan, Punit Singh Koura, Puxin Xu, Qing He, Qingxiao Dong, Ragavan Srinivasan, Raj Ganapathy, Ramon Calderer, Ricardo Silveira Cabral, Robert Stojnic, Roberta Raileanu, Rohan Maheswari, Rohit Girdhar, Rohit Patel, Romain Sauvestre, Ronnie Polidoro, Roshan Sumbaly, Ross Taylor, Ruan Silva, Rui Hou, Rui Wang, Saghar Hosseini, Sahana Chennabasappa, Sanjay Singh, Sean Bell, Seohyun Sonia Kim, Sergey Edunov, Shaoliang Nie, Sharan Narang, Sharath Raparthy, Sheng Shen, Shengye Wan, Shruti Bhosale, Shun Zhang, Simon Vandenhende, Soumya Batra, Spencer Whitman, Sten Sootla, Stephane Collot, Suchin Gururangan, Sydney Borodinsky, Tamar Herman, Tara Fowler, Tarek Sheasha, Thomas Georgiou, Thomas Scialom, Tobias Speckbacher, Todor Mihaylov, Tong Xiao, Ujjwal Karn, Vedanuj Goswami, Vibhor Gupta, Vignesh Ramanathan, Viktor Kerkez, Vincent Gonguet, Virginie Do, Vish Vogeti, Vítor Albiero, Vladan Petrovic, Weiwei Chu, Wenhan Xiong, Wenyin Fu, Whitney Meers, Xavier Martinet, Xiaodong Wang, Xiaofang Wang, Xiaoqing Ellen Tan, Xide Xia, Xinfeng Xie, Xuchao Jia, Xuewei Wang, Yaelle Goldschlag, Yashesh Gaur, Yasmine Babaei, Yi Wen, Yiwen Song, Yuchen Zhang, Yue Li, Yuning Mao, Zacharie Delpierre Coudert, Zheng Yan, Zhengxing Chen, Zoe Papakipos, Aaditya Singh, Aayushi Srivastava, Abha Jain, Adam Kelsey, Adam Shajnfeld, Adithya Gangidi, Adolfo Victoria, Ahuva Goldstand, Ajay Menon, Ajay Sharma, Alex Boesenberg, Alexei Baevski, Allie Feinstein, Amanda Kallet, Amit Sangani, Amos Teo, Anam Yunus, Andrei Lupu, Andres Alvarado, Andrew Caples, Andrew Gu, Andrew Ho, Andrew Poulton, Andrew Ryan, Ankit Ramchandani, Annie Dong, Annie Franco, Anuj Goyal, Aparajita Saraf, Arkabandhu Chowdhury, Ashley Gabriel, Ashwin Bharambe, Assaf Eisenman, Azadeh Yazdan, Beau James, Ben Maurer, Benjamin Leonhardi, Bernie Huang, Beth Loyd, Beto De Paola, Bhargavi Paranjape, Bing Liu, Bo Wu, Boyu Ni, Braden Hancock, Bram Wasti, Brandon Spence, Brani Stojkovic, Brian Gamido, Britt Montalvo, Carl Parker, Carly Burton, Catalina Mejia, Ce Liu, Changhan Wang, Changkyu Kim, Chao Zhou, Chester Hu, Ching-Hsiang Chu, Chris Cai, Chris Tindal, Christoph Feichtenhofer, Cynthia Gao, Damon Civin, Dana Beaty, Daniel Kreymer, Daniel Li, David Adkins, David Xu, Davide Testuggine, Delia David, Devi Parikh, Diana Liskovich, Didem Foss, Dingkang Wang, Duc Le, Dustin Holland, Edward Dowling, Eissa Jamil, Elaine Montgomery, Eleonora Presani, Emily Hahn, Emily Wood, Eric-Tuan Le, Erik Brinkman, Esteban Arcaute, Evan Dunbar, Evan Smothers, Fei Sun, Felix Kreuk, Feng Tian, Filippos Kokkinos, Firat Ozgenel, Francesco Caggioni, Frank Kanayet, Frank Seide, Gabriela Medina Florez, Gabriella Schwarz, Gada Badeer, Georgia Swee, Gil Halpern, Grant Herman, Grigory Sizov, Guangyi, Zhang, Guna Lakshminarayanan, Hakan Inan, Hamid Shojanazeri, Han Zou, Hannah Wang, Hanwen Zha, Haroun Habeeb, Harrison Rudolph, Helen Suk, Henry Aspegren, Hunter Goldman, Hongyuan Zhan, Ibrahim Damlaj, Igor Molybog, Igor Tufanov, Ilias Leontiadis, Irina-Elena Veliche, Itai Gat, Jake Weissman, James Geboski, James Kohli, Janice Lam, Japhet Asher, Jean-Baptiste Gaya, Jeff Marcus, Jeff Tang, Jennifer Chan, Jenny Zhen, Jeremy Reizenstein, Jeremy Teboul, Jessica Zhong, Jian Jin, Jingyi Yang, Joe Cummings, Jon Carvill, Jon Shepard, Jonathan McPhie, Jonathan Torres, Josh Ginsburg, Junjie Wang, Kai Wu, Kam Hou U, Karan Saxena, Kartikay Khandelwal, Katayoun Zand, Kathy Matosich, Kaushik Veeraraghavan, Kelly Michelena, Keqian Li, Kiran Jagadeesh, Kun Huang, Kunal Chawla, Kyle Huang, Lailin Chen, Lakshya Garg, Lavender A, Leandro Silva, Lee Bell, Lei Zhang, Liangpeng Guo, Licheng Yu, Liron Moshkovich, Luca Wehrstedt, Madian Khabsa, Manav Avalani, Manish Bhatt, Martynas Mankus, Matan Hasson, Matthew Lennie, Matthias Reso, Maxim Groshev, Maxim Naumov, Maya Lathi, Meghan Keneally, Miao Liu, Michael L. Seltzer, Michal Valko, Michelle Restrepo, Mihir Patel, Mik Vyatskov, Mikayel Samvelyan, Mike Clark, Mike Macey, Mike Wang, Miquel Jubert Hermoso, Mo Metanat, Mohammad Rastegari, Munish Bansal, Nandhini Santhanam, Natascha Parks, Natasha White, Navyata Bawa, Nayan Singhal, Nick Egebo, Nicolas Usunier, Nikhil Mehta, Nikolay Pavlovich Laptev, Ning Dong, Norman Cheng, Oleg Chernoguz, Olivia Hart, Omkar Salpekar, Ozlem Kalinli, Parkin Kent, Parth Parekh, Paul Saab, Pavan Balaji, Pedro Rittner, Philip Bontrager, Pierre Roux, Piotr Dollar, Polina Zvyagina, Prashant Ratanchandani, Pritish Yuvraj, Qian Liang, Rachad Alao, Rachel Rodriguez, Rafi Ayub, Raghotham Murthy, Raghu Nayani, Rahul Mitra, Rangaprabhu Parthasarathy, Raymond Li, Rebekkah Hogan, Robin Battey, Rocky Wang, Russ Howes, Ruty Rinott, Sachin Mehta, Sachin Siby, Sai Jayesh Bondu, Samyak Datta, Sara Chugh, Sara Hunt, Sargun Dhillon, Sasha Sidorov, Satadru Pan, Saurabh Mahajan, Saurabh Verma, Seiji Yamamoto, Sharadh Ramaswamy, Shaun Lindsay, Shaun Lindsay, Sheng Feng, Shenghao Lin, Shengxin Cindy Zha, Shishir Patil, Shiva Shankar, Shuqiang Zhang, Shuqiang Zhang, Sinong Wang, Sneha Agarwal, Soji Sajuyigbe, Soumith Chintala, Stephanie Max, Stephen Chen, Steve Kehoe, Steve Satterfield, Sudarshan Govindaprasad, Sumit Gupta, Summer Deng, Sungmin Cho, Sunny Virk, Suraj Subramanian, Sy Choudhury, Sydney Goldman, Tal Remez, Tamar Glaser, Tamara Best, Thilo Koehler, Thomas Robinson, Tianhe Li, Tianjun Zhang, Tim Matthews, Timothy Chou, Tzook Shaked, Varun Vontimitta, Victoria Ajayi, Victoria Montanez, Vijai Mohan, Vinay Satish Kumar, Vishal Mangla, Vlad Ionescu, Vlad Poenaru, Vlad Tiberiu Mihailescu, Vladimir Ivanov, Wei Li, Wenchen Wang, Wenwen Jiang, Wes Bouaziz, Will Constable, Xiaocheng Tang, Xiaojian Wu, Xiaolan Wang, Xilun Wu, Xinbo Gao, Yaniv Kleinman, Yanjun Chen, Ye Hu, Ye Jia, Ye Qi, Yenda Li, Yilin Zhang, Ying Zhang, Yossi Adi, Youngjin Nam, Yu, Wang, Yu Zhao, Yuchen Hao, Yundi Qian, Yunlu Li, Yuzi He, Zach Rait, Zachary DeVito, Zef Rosnbrick, Zhaoduo Wen, Zhenyu Yang, Zhiwei Zhao, and Zhiyu Ma. 2024. The llama 3 herd of models. *Preprint*, arXiv:2407.21783.

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

Cezary Kaliszyk and Florian Rabe. 2020. A survey of languages for formalizing mathematics. In *Intelligent Computer Mathematics: 13th International Conference, CICM 2020, Bertinoro, Italy, July 26–31, 2020, Proceedings*, page 138–156, Berlin, Heidelberg. Springer-Verlag.

Zenan Li, Yifan Wu, Zhaoyu Li, Xinming Wei, Xian Zhang, Fan Yang, and Xiaoxing Ma. 2024. Autoformalize mathematical statements by symbolic equivalence and semantic consistency. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*.

Pan Lu, Liang Qiu, Wenhao Yu, Sean Welleck, and Kai-Wei Chang. 2023. A survey of deep learning for mathematical reasoning. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 14605–14631, Toronto, Canada. Association for Computational Linguistics.

Jordan Meadows and André Freitas. 2023. Introduction to mathematical language processing: Informal proofs, word problems, and supporting tasks. *Transactions of the Association for Computational Linguistics*, 11:1162–1184.

Jordan Meadows, Marco Valentino, and Andre Freitas. 2023. Generating mathematical derivations with large language models. *arXiv preprint arXiv:2307.09998*.

Jordan Meadows, Marco Valentino, Damien Teney, and Andre Freitas. 2024. A symbolic framework for evaluating mathematical reasoning and generalisation with transformers. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 1505–1523, Mexico City, Mexico. Association for Computational Linguistics.

Swaroop Mishra, Matthew Finlayson, Pan Lu, Leonard Tang, Sean Welleck, Chitta Baral, Tanmay Rajpurohit, Oyvind Tafjord, Ashish Sabharwal, Peter Clark, and Ashwin Kalyan. 2022a. LILA: A unified benchmark for mathematical reasoning. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 5807–5832, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

Swaroop Mishra, Arindam Mitra, Neeraj Varshney, Bhavdeep Sachdeva, Peter Clark, Chitta Baral, and Ashwin Kalyan. 2022b. Numglue: A suite of fundamental yet challenging mathematical reasoning tasks. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 3505–3523.

Judit N. Moschkovich. 2003. What counts as mathematical discourse. *International Group for the Psychology of Mathematics Education*, 3:325–332.

Tobias Nipkow, Lawrence C. Paulson, and Markus Wenzel. 2002. *Isabelle/HOL — A Proof Assistant for Higher-Order Logic*, volume 2283 of *LNCS*. Springer.

OpenAI, :, Aaron Hurst, Adam Lerer, Adam P. Goucher, Adam Perelman, Aditya Ramesh, Aidan Clark, AJ Ostrow, Akila Welihinda, Alan Hayes, Alec Radford, Aleksander Mądry, Alex Baker-Whitcomb, Alex Beutel, Alex Borzunov, Alex Carney, Alex Chow, Alex Kirillov, Alex Nichol, Alex Paino, Alex Renzin, Alex Tachard Passos, Alexander Kirillov, Alexi Christakis, Alexis Conneau, Ali Kamali, Allan Jabri, Allison Moyer, Allison Tam, Amadou Crookes, Amin Tootoochian, Amin Tootoonchian, Ananya Kumar, Andrea Vallone, Andrej Karpathy, Andrew Braunstein, Andrew Cann, Andrew Codispoti, Andrew Galu, Andrew Kondrich, Andrew Tulloch, Andrey Mishchenko, Angela Baek, Angela Jiang, Antoine Pelisse, Antonia Woodford, Anuj Gosalia, Arka Dhar, Ashley Pantuliano, Avi Nayak, Avital Oliver, Barret Zoph, Behrooz Ghorbani, Ben Leimberger, Ben Rossen, Ben Sokolowsky, Ben Wang, Benjamin Zweig, Beth Hoover, Blake Samic, Bob McGrew, Bobby Spero, Bogo Giertler, Bowen Cheng, Brad Lightcap, Brandon Walkin, Brendan Quinn, Brian Guarraci, Brian Hsu, Bright Kellogg, Brydon Eastman, Camillo Lugaresi, Carroll Wainwright, Cary Bassin, Cary Hudson, Casey Chu, Chad Nelson, Chak Li, Chan Jun Shern, Channing Conger, Charlotte Barette, Chelsea Voss, Chen Ding, Cheng Lu, Chong Zhang, Chris Beaumont, Chris Hallacy, Chris Koch, Christian Gibson, Christina Kim, Christine Choi, Christine McLeavey, Christopher Hesse, Claudia Fischer, Clemens Winter, Coley Czarnecki, Colin Jarvis, Colin Wei, Constantin Koumouzelis, Dane Sherburn, Daniel Kappler, Daniel Levin, Daniel Levy, David Carr, David Farhi, David Mely, David Robinson, David Sasaki, Denny Jin, Dev Valladares, Dimitris Tsipras, Doug Li, Duc Phong Nguyen, Duncan Findlay, Edede Oiwoh, Edmund Wong, Ehsan Asdar, Elizabeth Proehl, Elizabeth Yang, Eric Antonow, Eric Kramer, Eric Peterson, Eric Sigler, Eric Wallace, Eugene Brevdo, Evan Mays, Farzad Khorasani, Felipe Petroski Such, Filippo Raso, Francis Zhang, Fred von Lohmann, Freddie Sulit, Gabriel Goh, Gene Oden, Geoff Salmon, Giulio Starace, Greg Brockman, Hadi Salman, Haiming Bao, Haitang Hu, Hannah Wong, Haoyu Wang, Heather Schmidt, Heather Whitney, Heewoo Jun, Hendrik Kirchner, Henrique Ponde de Oliveira Pinto, Hongyu Ren, Huiwen Chang, Hyung Won Chung, Ian Kivlichan, Ian O'Connell, Ian O'Connell, Ian Osband, Ian Silber, Ian Sohl, Ibrahim Okuyucu, Ikai Lan, Ilya Kostrikov, Ilya Sutskever, Ingmar Kanitscheider, Ishaan Gulrajani, Jacob Coxon, Jacob Menick, Jakub Pachocki, James Aung, James Betker, James Crooks, James Lennon, Jamie Kiros, Jan Leike, Jane Park, Jason Kwon, Jason Phang, Jason Teplitz, Jason Wei, Jason Wolfe, Jay Chen, Jeff Harris, Jenia Varavva, Jessica Gan Lee, Jessica Shieh, Ji Lin, Jiahui Yu, Jiayi Weng, Jie Tang, Jieqi Yu, Joanne Jang, Joaquin Quinonero Candela, Joe Beutler, Joe Landers, Joel Parish, Johannes Heidecke, John Schulman, Jonathan Lachman, Jonathan McKay, Jonathan Uesato, Jonathan Ward, Jong Wook Kim, Joost Huizinga, Jordan Sitkin, Jos Kraaijeveld, Josh Gross, Josh Kaplan, Josh Snyder, Joshua Achiam, Joy Jiao, Joyce Lee, Juntang Zhuang, Justyn Harriman, Kai Fricke, Kai Hayashi, Karan Singhal, Katy Shi, Kavin Karthik, Kayla Wood, Kendra Rimbach, Kenny Hsu, Kenny Nguyen, Keren Gu-Lemberg, Kevin Button, Kevin Liu, Kiel Howe, Krithika Muthukumar, Kyle Luther, Lama Ahmad, Larry Kai, Lauren Itow, Lauren Workman, Leher Pathak, Leo Chen, Li Jing, Lia Guy, Liam Fedus, Liang Zhou, Lien Mamitsuka, Lilian Weng, Lindsay McCallum, Lindsey Held, Long Ouyang, Louis Feuvrier, Lu Zhang, Lukas Kondraciuk, Lukasz Kaiser, Luke Hewitt, Luke Metz, Lyric Doshi, Mada Aflak, Maddie Simens, Madelaine Boyd, Madeleine Thompson, Marat Dukhan, Mark Chen, Mark Gray, Mark Hudnall, Marvin Zhang, Marwan Aljubeh, Mateusz Litwin, Matthew Zeng, Max Johnson, Maya Shetty, Mayank Gupta, Meghan Shah, Mehmet Yatbaz, Meng Jia Yang, Mengchao Zhong, Mia Glaese, Mianna Chen, Michael Janner, Michael Lampe, Michael Petrov, Michael Wu, Michele Wang, Michelle Fradin, Michelle Pokrass, Miguel Castro, Miguel Oom Temudo de Castro, Mikhail Pavlov, Miles Brundage, Miles Wang, Minal Khan, Mira Murati, Mo Bavarian, Molly Lin, Murat Yesildal, Nacho Soto, Natalia Gimelshein, Natalie Cone, Natalie Staudacher, Natalie Summers, Natan LaFontaine, Neil Chowdhury, Nick Ryder, Nick Stathas, Nick Turley, Nik Tezak, Niko Felix, Nithanth Kudige, Nitish Keskar, Noah Deutsch, Noel Bundick, Nora Puckett, Ofir Nachum, Ola Okelola, Oleg Boiko, Oleg Murk, Oliver Jaffe, Olivia Watkins, Olivier Godement, Owen Campbell-Moore, Patrick Chao, Paul McMillan, Pavel Belov, Peng Su, Peter Bak, Peter Bakkum, Peter Deng, Peter Dolan, Peter Hoeschele, Peter Welinder, Phil Tillet, Philip Pronin, Philippe Tillet, Prafulla Dhariwal, Qiming Yuan, Rachel Dias, Rachel Lim, Rahul Arora, Rajan Troll, Randall Lin, Rapha Gontijo Lopes, Raul Puri, Reah Miyara, Reimar Leike, Renaud Gaubert, Reza Zamani, Ricky Wang, Rob Donnelly, Rob Honsby, Rocky Smith, Rohan Sahai, Rohit Ramchandani, Romain Huet, Rory Carmichael, Rowan Zellers, Roy Chen, Ruby Chen, Ruslan Nigmatullin, Ryan Cheu, Saachi Jain, Sam Altman, Sam Schoenholz, Sam Toizer, Samuel Miserendino, Sandhini Agarwal, Sara Culver, Scott Ethersmith, Scott Gray, Sean Grove, Sean Metzger, Shamez Hermani, Shantanu Jain, Shengjia Zhao, Sherwin Wu, Shino Jomoto, Shirong Wu, Shuaiqi, Xia, Sonia Phene, Spencer Papay, Srinivas Narayanan, Steve Coffey, Steve Lee, Stewart Hall, Suchir Balaji, Tal Broda, Tal Stramer, Tao Xu, Tarun Gogineni, Taya Christianson, Ted Sanders, Tejal Patwardhan, Thomas Cunninghman, Thomas Degry, Thomas Dimson, Thomas Raoux, Thomas Shadwell, Tianhao Zheng, Todd Underwood, Todor Markov, Toki Sherbakov, Tom Rubin, Tom Stasi, Tomer Kaftan, Tristan Heywood, Troy Peterson, Tyce Walters, Tyna Eloundou, Valerie Qi, Veit Moeller, Vinnie Monaco, Vishal Kuo, Vlad Fomenko, Wayne Chang, Weiyi Zheng, Wenda Zhou, Wesam Manassra, Will Sheu, Wojciech Zaremba, Yash Patil, Yilei Qian, Yongjik Kim, Youlong Cheng, Yu Zhang, Yuchen He, Yuchen Zhang, Yujia Jin, Yunxing Dai, and Yury Malkov. 2024a. Gpt-4o system card. *Preprint*, arXiv:2410.21276.

OpenAI, Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, Red Avila, Igor Babuschkin, Suchir Balaji, Valerie Balcom, Paul Baltescu, Haiming Bao, Mohammad Bavarian, Jeff Belgum, Irwan Bello, Jake Berdine, Gabriel Bernadett-Shapiro, Christopher Berner, Lenny Bogdonoff, Oleg Boiko, Madelaine Boyd, Anna-Luisa Brakman, Greg Brockman, Tim Brooks, Miles Brundage, Kevin Button, Trevor Cai, Rosie Campbell, Andrew Cann, Brittany Carey, Chelsea Carlson, Rory Carmichael, Brooke Chan, Che Chang, Fotis Chantzis, Derek Chen, Sully Chen, Ruby Chen, Jason Chen, Mark Chen, Ben Chess, Chester Cho, Casey Chu, Hyung Won Chung, Dave Cummings, Jeremiah Currier, Yunxing Dai, Cory Decareaux, Thomas Degry, Noah Deutsch, Damien Deville, Arka Dhar, David Dohan, Steve Dowling, Sheila Dunning, Adrien Ecoffet, Atty Eleti, Tyna Eloundou, David Farhi, Liam Fedus, Niko Felix, Simón Posada Fishman, Juston Forte, Isabella Fulford, Leo Gao, Elie Georges, Christian Gibson, Vik Goel, Tarun Gogineni, Gabriel Goh, Rapha Gontijo-Lopes, Jonathan Gordon, Morgan Grafstein, Scott Gray, Ryan Greene, Joshua Gross, Shixiang Shane Gu, Yufei Guo, Chris Hallacy, Jesse Han, Jeff Harris, Yuchen He, Mike Heaton, Johannes Heidecke, Chris Hesse, Alan Hickey, Wade Hickey, Peter Hoeschele, Brandon Houghton, Kenny Hsu, Shengli Hu, Xin Hu, Joost Huizinga, Shantanu Jain, Shawn Jain, Joanne Jang, Angela Jiang, Roger Jiang, Haozhun Jin, Denny Jin, Shino Jomoto, Billie Jonn, Heewoo Jun, Tomer Kaftan, Łukasz Kaiser, Ali Kamali, Ingmar Kanitscheider, Nitish Shirish Keskar, Tabarak Khan, Logan Kilpatrick, Jong Wook Kim, Christina Kim, Yongjik Kim, Jan Hendrik Kirchner, Jamie Kiros, Matt Knight, Daniel Kokotajlo, Łukasz Kondraciuk, Andrew Kondrich, Aris Konstantinidis, Kyle Kosic, Gretchen Krueger, Vishal Kuo, Michael Lampe, Ikai Lan, Teddy Lee, Jan Leike, Jade Leung, Daniel Levy, Chak Ming Li, Rachel Lim, Molly Lin, Stephanie Lin, Mateusz Litwin, Theresa Lopez, Ryan Lowe, Patricia Lue, Anna Makanju, Kim Malfacini, Sam Manning, Todor Markov, Yaniv Markovski, Bianca Martin, Katie Mayer, Andrew Mayne, Bob McGrew, Scott Mayer McKinney, Christine McLeavey, Paul McMillan, Jake McNeil, David Medina, Aalok Mehta, Jacob Menick, Luke Metz, Andrey Mishchenko, Pamela Mishkin, Vinnie Monaco, Evan Morikawa, Daniel Mossing, Tong Mu, Mira Murati, Oleg Murk, David Mély, Ashvin Nair, Reiichiro Nakano, Rajeev Nayak, Arvind Neelakantan, Richard Ngo, Hyeonwoo Noh, Long Ouyang, Cullen O'Keefe, Jakub Pachocki, Alex Paino, Joe Palermo, Ashley Pantuliano, Giambattista Parascandolo, Joel Parish, Emy Parparita, Alex Passos, Mikhail Pavlov, Andrew Peng, Adam Perelman, Filipe de Avila Belbute Peres, Michael Petrov, Henrique Ponde de Oliveira Pinto, Michael, Pokorny, Michelle Pokrass, Vitchyr H. Pong, Tolly Powell, Alethea Power, Boris Power, Elizabeth Proehl, Raul Puri, Alec Radford, Jack Rae, Aditya Ramesh, Cameron Raymond, Francis Real, Kendra Rimbach, Carl Ross, Bob Rotsted, Henri Roussez, Nick Ryder, Mario Saltarelli, Ted Sanders, Shibani Santurkar, Girish Sastry, Heather Schmidt, David Schnurr, John Schulman, Daniel Selsam, Kyla Sheppard, Toki Sherbakov, Jessica Shieh, Sarah Shoker, Pranav Shyam, Szymon Sidor, Eric Sigler, Maddie Simens, Jordan Sitkin, Katarina Slama, Ian Sohl, Benjamin Sokolowsky, Yang Song, Natalie Staudacher, Felipe Petroski Such, Natalie Summers, Ilya Sutskever, Jie Tang, Nikolas Tezak, Madeleine B. Thompson, Phil Tillet, Amin Tootoonchian, Elizabeth Tseng, Preston Tuggle, Nick Turley, Jerry Tworek, Juan Felipe Cerón Uribe, Andrea Vallone, Arun Vijayvergiya, Chelsea Voss, Carroll Wainwright, Justin Jay Wang, Alvin Wang, Ben Wang, Jonathan Ward, Jason Wei, CJ Weinmann, Akila Welihinda, Peter Welinder, Jiayi Weng, Lilian Weng, Matt Wiethoff, Dave Willner, Clemens Winter, Samuel Wolrich, Hannah Wong, Lauren Workman, Sherwin Wu, Jeff Wu, Michael Wu, Kai Xiao, Tao Xu, Sarah Yoo, Kevin Yu, Qiming Yuan, Wojciech Zaremba, Rowan Zellers, Chong Zhang, Marvin Zhang, Shengjia Zhao, Tianhao Zheng, Juntang Zhuang, William Zhuk, and Barret Zoph. 2024b. Gpt-4 technical report. *Preprint*, arXiv:2303.08774.

Felix Petersen, Moritz Schubotz, Andre Greiner-Petter, and Bela Gipp. 2023. Neural machine translation for mathematical formulae. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 11534–11550, Toronto, Canada. Association for Computational Linguistics.

Xin Quan, Marco Valentino, Louise Dennis, and Andre Freitas. 2024a. Enhancing ethical explanations of large language models through iterative symbolic refinement. In *Proceedings of the 18th Conference of the European Chapter of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1–22, St. Julian's, Malta. Association for Computational Linguistics.

Xin Quan, Marco Valentino, Louise A. Dennis, and Andre Freitas. 2024b. Verification and refinement of natural language explanations through LLM-symbolic theorem proving. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 2933–2958, Miami, Florida, USA. Association for Computational Linguistics.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, Y. K. Li, Y. Wu, and Daya Guo. 2024. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. *Preprint*, arXiv:2402.03300.

Guillem Tarrach, Albert Q. Jiang, Daniel Raggi, Wenda Li, and Mateja Jamnik. 2024. More details, please: Improving autoformalization with more detailed proofs. In *AI for Math Workshop @ ICML 2024*.

Marco Valentino, Deborah Ferreira, Mokanarangan Thayaparan, André Freitas, and Dmitry Ustalov. 2022. TextGraphs 2022 shared task on natural language premise selection. In *Proceedings of TextGraphs-16: Graph-based Methods for Natural Language Processing*, pages 105–113, Gyeongju, Republic of Korea. Association for Computational Linguistics.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, brian ichter, Fei Xia, Ed H. Chi, Quoc V Le, and Denny Zhou. 2022. Chain of thought prompting elicits reasoning in large language models. In *Advances in Neural Information Processing Systems*.

Sean Welleck, Jiacheng Liu, Ronan Le Bras, Hannaneh Hajishirzi, Yejin Choi, and Kyunghyun Cho. 2021. Naturalproofs: Mathematical theorem proving in natural language. In *Thirty-fifth Conference on Neural Information Processing Systems Datasets and Benchmarks Track (Round 1)*.

Yuhuai Wu, Albert Qiaochu Jiang, Wenda Li, Markus Norman Rabe, Charles E Staats, Mateja Jamnik, and Christian Szegedy. 2022. Autoformalization with large language models. In *Advances in Neural Information Processing Systems*.

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

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*.

A Case Study for Formal Definition Grounding

The following example shows an example of using GPT-4o in a zero-shot setting to formalize the definition of Bradley–Terry model[3].

Definition of Bradley–Terry model: Given a pair of items i and j drawn from some population, the Bradley–Terry model estimates the probability that the pairwise comparison turns out true, as

$$\Pr(i > j) = \frac{p_i}{p_i + p_j}$$

where p_i is a positive real-valued score assigned to individual i.

Generated Formal Code:

```
theory test
imports Main
begin
definition bradley_terry :: "real ⇒ real
  ⇒ real" where "bradley_terry p_i p_j =
    p_i / (p_i + p_j)"
end
```

The preamble in the generated formal code is "Main". However, "Main" does not contain the formalization of "real", making the formal code invalid. After applying Post-FDG, the preamble is updated to "HOL.Real", and the formal code becomes valid. One might suggest creating a universal preamble that imports all source files from the library, applying this common preamble to solve such issues. However, this approach would not align with how a human expert would perform formalization. This failure to identify the correct preambles exposes limitations in the autoformalization capabilities of LLMs. Another issue, which is outside the scope of this paper but an important future direction, is that while Post-FDG can correct the formal code, the semantics of the generated code still do not fully match the original natural

[3] https://en.wikipedia.org/wiki/Bradley%E2%80%93Terry_model

Category	Reasons
SYN	1. **Invalid Symbol Format.** Isabelle uses symbols like "\<Rightarrow>" to represent "\Rightarrow (⇒)" in LaTeX. GPT-4o does not strictly follow this behaviour. A symbol in its formalized code starting with "\<" can miss ">" at the end so that the relevant symbol is not valid. 2. **Confusion of Mapping between LaTeX Mathematical Symbols and Isabelle Symbols.** Not all natural language symbols in LaTeX have a similar corresponding version in Isabelle symbols. In natural language mathematics we use different mathematical fonts such as "\mathcal ($\mathcal{A}$)" to distinguish items. Isabelle uses "\<A>" to represent this LaTeX symbol. However, GPT-4o would pretend the existence of a symbol named \<mathcal> and use it for autoformalization. 3. **Unaware of Name Conflict.** Some keywords such as "instance" are reserved by Isabelle/HOL and they cannot be used as the name of a new item. 4. **Incorrect Stylistic Usage of Symbols or Operators.** Some symbols or operators require specific usage which is not in the same style as in natural language. The incorrect usage of them in formalized code generated by GPT-4o can lead to syntax errors.
UDF	1. **Items not defined.** Formalization requires every mentioned item to be clearly defined in the local context or preambles. For one piece of generated formal code, GPT-4o could refer to items that are not defined in both sources.
TUF	1. **Mismatch between Types in Definition and Types in Actual Usage.** There are some operators or functions which have been clearly defined about the types of their operands or parameters. When using these operators or functions, the types of actual operands or parameters need to match the types in the definitions exactly. GPT-4o would produce mismatched types in the formalized codes and introduce TUF errors.

Table 5: Reasons of failure in each error category during autoformalization with GPT-4o.

language version. For instance, the term "probability" does not appear in the formal code, and the phrase "p_i" is a positive real number" is omitted. The challenge of measuring semantic consistency between the generated formal code and its corresponding natural language version remains an open problem.

B Examples of Incorrect Formal Codes

In this section, we provide some examples of incorrect formal codes generated by GPT-4o to support our summarized reasons in Table 5. All examples of definitions are from Def_Wiki development set.

Example 1 is about autoformalization on the definition of Rademacher Complexity[4]. In this example, the missing of ">" in "\<sigma"leads to a SYN error, which contributes to reason 1 of SYN category. In addition, this piece of formal code uses "real" but "real" is not imported by "imports Main". This problem contributes to reason 1 of UDF category.

Example 1:
Definition of Rademacher Complexity: Given a set $A \subseteq \mathbb{R}^m$, the Rademacher complexity of A is defined as follows:

$$\mathrm{Rad}(A) := \frac{1}{m}\mathbb{E}_\sigma\left[\sup_{a\in A}\sum_{i=1}^{m}\sigma_i a_i\right]$$

where $\sigma_1, \sigma_2, \ldots, \sigma_m$ are independent random variables drawn from the Rademacher distribution (i.e. $\Pr(\sigma_i = +1) = \Pr(\sigma_i = -1) = 1/2$ for $i = 1, 2, \ldots, m$), and $a = (a_1, \ldots, a_m)$.
Generated Formal Code:

```
theory test
imports Main
begin

definition rademacher_distribution :: "nat
  ⇒ real pmf" where "rademacher_distribution
    m = pmf_of_set {x. x = 1 ∨ x = -1}"

definition rademacher_complexity :: "(real
  vector set) ⇒ nat ⇒ real" where
  "rademacher_complexity A m =
    (1 / real m) * (expectation (pmf_of_set
    {x. x = 1 ∨ x = -1}) (λσ. Sup {sum (λi.
    \<sigma i * a i) {1..m} | a. a ∈ A}))"

end
```

Example 2 is about autoformalization on the definition of Algorithm Selection[5]. In this example, the problem of missing ">" and using "real" also occurs. Even if missing ">" is solved, "\<mathcal> is not a valid symbol in Isabelle. This problem

[4] https://en.wikipedia.org/wiki/Rademacher_complexity

[5] https://en.wikipedia.org/wiki/Algorithm_selection

contributes to reason 2 of SYN category. In addition, “instance” is a reserved keyword in Isabelle so it cannot be used in “type_synonym”. This contributes to reason 3 of SYN category.

Example 2:
Definition of Algorithm Selection: Given a portfolio $\mathcal{P}$ of algorithms $\mathcal{A} \in \mathcal{P}$, a set of instances $i \in \mathcal{I}$ and a cost metric $m : \mathcal{P} \times \mathcal{I} \to \mathbb{R}$, the algorithm selection problem consists of finding a mapping $s : \mathcal{I} \to \mathcal{P}$ from instances $\mathcal{I}$ to algorithms $\mathcal{P}$ such that the cost $\sum_{i \in \mathcal{I}} m(s(i), i)$ across all instances is optimized.
Generated Formal Code:

```
theory test
imports Main
begin

type_synonym 'a instance = 'a
type_synonym 'a algorithm = 'a
type_synonym cost_metric = "'a algorithm
  ⇒ 'a instance ⇒ real"

definition algorithm_selection ::
  "'a algorithm set ⇒ 'a instance set
  ⇒ cost_metric ⇒ ('a instance
  ⇒ 'a algorithm) ⇒ real" where
  "algorithm_selection \<mathcal{P}
    \<mathcal{I} m s = (∑i ∈ \<mathcal{I}.
    m (s i) i)"

end
```

Example 3 is about autoformalization on the definition of Set-Family Growth Function[6]. In this example, there is no SYN or UDF errors. The error occurs in the definition of “growth_function”, where “card ()” is used on the right side of operator “⟶”. However, the left and right sides of operator “⟶” are explicitly defined as type “bool” and the type of “card ()” is explicitly defined as “nat” in the Isabelle/HOL library. This incorrect usage of them causes a type mismatch and leads to TUF error.

[6] https://en.wikipedia.org/wiki/Growth_function

Example 3:
Definition of Set-Family Growth Function: Let H be a set family (a set of sets) and C a set. Their intersection is defined as the following set-family: $H \cap C := \{h \cap C \mid h \in H\}$. The intersection-size (also called the index) of H with respect to C is $|H \cap C|$. The growth function measures the size of $H \cap C$ as a function of $|C|$. Formally: $\mathrm{Growth}(H, m) := \max_{C:|C|=m} |H \cap C|$.
Generated Formal Code:

```
theory test
imports Main
begin
definition intersection_set_family ::
  "'a set set ⇒ 'a set ⇒ 'a set set"
  (infixl "∩\<^sub>H" 70) where
  "H ∩\<^sub>H C = {h ∩ C | h. h ∈ H}"

definition intersection_size ::
  "'a set set ⇒ 'a set ⇒ nat" where
  "intersection_size H C = card (
    H ∩\<^sub>H C)"

definition growth_function :: "'a set set ⇒
  nat ⇒ nat" where
  "growth_function H m = (SUP C. card C =
    m ∧ finite C ∧ (∀h ∈ H. finite h)
    ∧ finite H⟶card (H ∩\<^sub>H C))"
end
```

C Prompts and Detailed Results

The prompts used for the estimation of dataset statistics are provided in Table 6. The instructions used in the prompts of experiments are provided in Table 7. Detailed numbers of autoformalization results on miniF2F test set, Def_Wiki test set and Def_ArXiv are provided in Table 8, 9, 10, respectively. Symbolic refinement results and Post-FDG results on Def_Wiki test set are provided in Table 11 and Table 12, respectively.

Purpose	Content
Mathematical Objects	Given the following statement written in LaTeX: {{latex}} How many mathematical objects excluding explicit numbers and variables are mentioned directly in this statement? You can think it step by step. Give me the final number as NUMBER={the number}
Mathematical Formulae	Given the following statement written in LaTeX: {{latex}} How many mathematical formulae are mentioned directly in this statement? You can think it step by step. Give me the final number as NUMBER={the number}

Table 6: Prompts for the estimation of dataset statistics.

Instruction	Content
General	You are an expert in *machine learning* and *formal language Isabelle/HOL*. Given the following definition in LaTeX: {{latex}}, your task is to provide the formal code of this definition in Isabelle/HOL. The following text might contain some preliminaries to explain the given definition: {{preliminary}}. In case that you need to import any necessary dependent theory files, you should not import any fake theory files.
Stylistic	To represent the math symbols, you must use the textual full name of symbols in Isabelle instead of direct symbols. For example you should use \<Rightarrow> instead of $\Rightarrow$, \<lambda> instead of λ.
Output	Give the results directly without any additional explanations.
Refinement	**Plain**: For your reference, there are some previous formal codes generated by you: {{previous}}. You can choose to refine this piece of code for your task. **Binary**: For your reference, there are some previous formal codes generated by you: {{previous}}. The syntactic correctness for this piece of code is: {{correctness}}. You can choose to refine this piece of code for your task. **Detailed**: For your reference, there are some previous formal codes generated by you: {{previous}}. The provided code might have some errors according to the Isabelle prover. The error details and where the error code is located in the code are: {{error_details}}. You should refine this piece of code for your task.
SYN	You should make sure that every symbol you use is a valid Isabelle symbol. If an Isabelle symbol starts with \<, then it must end with >. Isabelle reserves some words as keywords. You should be careful with this and avoid to use them to define new names. You should make sure that the usage of symbols and operators is correct in your final output as the incorrect usage will lead to syntax errors.
UDF	You should make sure that every item you mentioned in your code has a clear reference either in the local context or the theory files that you decide to import.
TUF	You should make sure that in your code, the types of operands of operators or the types of parameters of functions match the types in their definitions exactly. Failure to maintain such compatibility will lead to type mismatch errors.
Include Formal Definition Codes	**Soft**: You can use the following Isabelle/HOL codes to support your task: {{formal_defs}} but you should not restate these codes in your final output. You need to formalize everything that is not provided in the given code. In this case, you should assume that you can only use things from HOL.Main. You only need to provide the main body of formal codes for the given definition. You may not import any theory files. **Hard**: The following Isabelle/HOL codes define some mathematical concepts which might be related to your task: {{formal_defs}}. If a mathematical concept in your task has been defined in the above codes, you are required to use this version of formal codes but you should not restate these codes in your final output. You need to formalize everything that is not provided in the given code. In this case, you should assume that you can only use things from HOL.Main. You only need to provide the main body of formal codes for the given definition. You may not import any theory files.

Table 7: Instructions used in prompts.

Prompt Strategy	Preamble	Pass↑	FEO↑	TRO↓	IVI↓	SYN↓	UDF↓	TUF↓
DeepSeekMath-7B								
ZS	Direct	3.28	12.79	18.44	0.00	50.00	14.34	9.43
	Post-FDG	12.30	23.60	15.98	0.00	47.13	1.23	9.02
(ZS) + Binary	Direct	2.05	6.73	2.46	0.00	79.91	5.33	2.05
	Post-FDG	4.10	9.39	2.46	0.00	80.33	0.41	1.23
(ZS) + Detailed	Direct	3.28	10.03	5.74	0.00	70.49	10.66	4.10
	Post-FDG	5.74	15.57	5.74	0.00	69.67	0.82	0.41
(ZS) + Detailed + CR-All	Direct	3.28	9.11	6.15	0.00	73.77	6.15	3.28
	Post-FDG	5.33	13.08	6.15	0.00	72.95	0.41	3.28
Llama3-8B								
ZS	Direct	4.92	20.70	4.51	0.41	29.51	38.52	18.85
	Post-FDG	10.66	31.17	4.92	0.00	28.69	20.08	21.31
(ZS) + Binary	Direct	3.69	20.52	3.28	0.41	33.20	39.75	20.49
	Post-FDG	9.43	30.57	3.69	0.00	31.97	22.95	22.13
(ZS) + Detailed	Direct	4.10	24.33	3.69	0.82	29.51	35.25	18.44
	Post-FDG	9.02	33.36	4.10	0.00	27.46	18.44	22.13
(ZS) + Detailed + CR-All	Direct	4.92	24.16	6.97	0.82	27.46	35.25	20.08
	Post-FDG	9.43	32.41	7.79	0.00	27.46	18.85	22.54
GPT-4o								
ZS	Direct	25.41	48.90	1.23	1.23	6.15	23.77	7.38
	Post-FDG	67.21	81.88	0.00	0.00	3.28	2.87	5.33
ZS + CR-SYN	Direct	24.18	45.31	2.46	0.00	9.02	27.46	7.79
	Post-FDG	52.46	73.96	0.41	0.00	7.79	3.69	3.69
ZS + CR-UDF	Direct	25.82	50.75	2.05	2.46	6.56	22.54	6.97
	Post-FDG	61.48	80.41	0.41	0.00	5.33	2.87	2.87
ZS + CR-TUF	Direct	27.87	50.62	2.05	1.64	5.33	26.64	5.74
	Post-FDG	54.10	78.79	0.00	0.00	3.28	4.10	2.87
(ZS)	Direct	25.41	53.15	1.64	1.23	6.56	22.13	7.79
	Post-FDG	67.21	84.05	0.00	0.00	3.28	2.46	4.92
(ZS) + Binary	Direct	29.10	53.90	2.05	1.23	6.15	21.72	8.20
	Post-FDG	67.21	83.60	0.00	0.00	4.10	2.05	4.92
(ZS) + Detailed	Direct	37.30	63.28	2.05	1.23	5.74	9.02	8.61
	Post-FDG	83.61	91.47	0.00	0.00	2.05	0.82	3.28
(ZS) + CR-SYN	Direct	25.41	52.72	2.05	1.23	5.74	22.13	8.61
	Post-FDG	67.21	83.73	0.00	0.00	2.87	2.46	5.74
(ZS) + CR-UDF	Direct	26.64	54.06	1.64	1.23	6.15	21.72	6.97
	Post-FDG	67.21	83.78	0.00	0.00	3.69	2.05	4.92
(ZS) + CR-TUF	Direct	25.41	51.18	2.46	1.23	6.56	24.18	7.38
	Post-FDG	67.21	83.94	0.00	0.00	3.28	2.87	4.10
(ZS) + Detailed + CR-SYN	Direct	38.52	64.42	2.05	1.23	7.79	8.20	7.79
	Post-FDG	82.79	90.32	0.00	0.00	3.28	0.82	2.05
(ZS) + Detailed + CR-UDF	Direct	38.11	63.95	2.05	2.46	5.74	6.56	6.97
	Post-FDG	82.38	90.48	0.00	0.00	2.46	1.23	2.87
(ZS) + Detailed + CR-TUF	Direct	41.39	64.76	3.28	1.23	6.15	11.07	6.15
	Post-FDG	83.20	90.71	0.00	0.00	2.87	1.64	2.05
(ZS) + Detailed + CR-All	Direct	38.52	65.73	2.05	1.23	6.15	5.74	7.79
	Post-FDG	81.97	90.65	0.00	0.00	2.46	0.41	2.46

Table 8: Error results on miniF2F test set.

Prompt Strategy	Preamble	Pass↑	FEO↑	TRO↓	IVI↓	SYN↓	UDF↓	TUF↓
DeepSeekMath-7B								
ZS	Direct	10.87	17.75	34.78	2.17	30.43	26.09	2.17
	Post-FDG	26.09	30.98	34.78	0.00	21.74	10.87	13.04
(ZS) + Binary	Direct	6.52	7.73	8.70	0.00	69.57	21.74	2.17
	Post-FDG	10.87	12.56	8.70	0.00	65.22	15.22	6.52
(ZS) + Detailed	Direct	10.87	13.27	15.22	2.17	43.48	34.78	6.52
	Post-FDG	26.09	29.21	13.04	0.00	36.96	17.39	19.57
(ZS) + Detailed + CR-All	Direct	4.35	7.66	13.04	2.17	47.83	32.61	8.70
	Post-FDG	17.39	21.43	13.04	0.00	41.30	15.22	21.74
Llama3-8B								
ZS	Direct	0.00	2.80	0.00	23.91	56.52	32.61	4.35
	Post-FDG	0.00	2.80	21.74	0.00	58.70	23.91	15.22
(ZS) + Binary	Direct	2.17	3.71	0.00	26.09	52.17	30.43	2.17
	Post-FDG	0.00	1.53	23.91	0.00	56.52	28.26	13.04
(ZS) + Detailed	Direct	2.17	3.80	0.00	26.09	50.00	30.43	6.52
	Post-FDG	4.35	5.98	23.91	0.00	52.17	26.09	15.22
(ZS) + Detailed + CR-All	Direct	2.17	3.71	0.00	26.09	52.17	32.61	4.35
	Post-FDG	2.17	3.71	23.91	0.00	54.35	23.91	15.22
GPT-4o								
ZS	Direct	10.87	16.12	8.70	8.70	19.57	50.00	13.04
	Post-FDG	34.78	42.56	6.52	0.00	30.43	17.39	23.91
ZS + CR-SYN	Direct	10.87	15.18	8.70	2.17	15.22	58.70	13.04
	Post-FDG	34.78	40.27	8.70	0.00	28.26	13.04	26.09
ZS + CR-UDF	Direct	2.17	11.59	6.52	6.52	19.57	60.87	19.57
	Post-FDG	30.43	42.66	2.17	0.00	34.78	23.91	23.91
ZS + CR-TUF	Direct	8.70	14.55	8.70	6.52	21.74	56.52	15.22
	Post-FDG	30.43	40.51	6.52	0.00	34.78	17.39	28.26
(ZS)	Direct	10.87	16.21	8.70	8.70	19.57	50.00	13.04
	Post-FDG	39.13	47.23	6.52	0.00	28.26	15.22	23.91
(ZS) + Binary	Direct	13.04	18.30	8.70	6.52	17.39	50.00	13.04
	Post-FDG	39.13	48.00	6.52	0.00	26.09	8.70	28.26
(ZS) + Detailed	Direct	19.57	23.46	8.70	8.70	10.87	47.83	10.87
	Post-FDG	43.48	50.13	6.52	0.00	21.74	10.87	23.91
(ZS) + CR-SYN	Direct	10.87	16.12	8.70	8.70	17.39	52.17	13.04
	Post-FDG	36.96	44.97	6.52	0.00	30.43	15.22	23.91
(ZS) + CR-UDF	Direct	10.87	16.12	8.70	8.70	19.57	50.00	13.04
	Post-FDG	36.96	44.97	6.52	0.00	30.43	15.22	23.91
(ZS) + CR-TUF	Direct	10.87	16.21	8.70	8.70	21.74	47.83	13.04
	Post-FDG	36.96	45.06	6.52	0.00	32.61	15.22	21.74
(ZS + Detailed) + Detailed	Direct	19.57	24.09	8.70	8.70	13.04	43.48	10.87
	Post-FDG	43.48	50.32	6.52	0.00	19.57	8.70	26.09
(ZS) + Detailed + CR-SYN	Direct	21.74	25.63	8.70	10.87	10.87	41.30	8.70
	Post-FDG	45.65	52.31	6.52	0.0	21.74	8.70	21.74
(ZS) + Detailed + CR-UDF	Direct	17.39	21.83	8.70	13.04	17.39	39.13	8.70
	Post-FDG	43.48	50.24	6.52	0.0	21.74	10.87	21.74
(ZS) + Detailed + CR-TUF	Direct	19.57	23.46	8.70	8.70	17.39	43.48	8.70
	Post-FDG	45.65	52.31	6.52	0.0	23.91	10.87	19.57
(ZS) + Detailed + CR-All	Direct	21.74	25.63	8.70	8.70	10.87	43.48	13.04
	Post-FDG	43.48	50.13	6.52	0.00	21.74	10.87	23.91

Table 9: Error results on Def_Wiki test set.

Prompt Strategy	Preamble	Pass↑	FEO↑	TRO↓	IVI↓	SYN↓	UDF↓	TUF↓
DeepSeekMath-7B								
ZS	Direct	13.33	14.69	16.67	0.00	40.00	36.67	13.33
	Post-FDG	16.67	18.02	13.33	0.00	43.33	30.00	16.67
(ZS) + Binary	Direct	3.33	3.33	6.67	0.00	66.67	33.33	3.33
	Post-FDG	6.67	7.41	3.33	0.00	70.00	23.33	10.00
(ZS) + Detailed	Direct	6.67	7.36	13.33	0.00	46.67	43.33	13.33
	Post-FDG	13.33	14.02	10.00	0.00	46.67	33.33	20.00
(ZS) + Detailed + CR-All	Direct	6.67	7.59	13.33	0.00	46.67	43.33	13.33
	Post-FDG	13.33	14.26	10.00	0.00	46.67	33.33	20.00
Llama3-8B								
ZS	Direct	0.00	2.67	0.00	13.33	70.00	40.00	6.67
	Post-FDG	0.00	2.67	13.33	0.00	66.67	26.67	20.00
(ZS) + Binary	Direct	3.33	5.83	0.00	20.00	60.00	33.33	6.67
	Post-FDG	3.33	5.83	20.00	0.00	60.00	26.67	16.67
(ZS) + Detailed	Direct	0.00	1.41	0.00	20.00	63.33	33.33	6.67
	Post-FDG	0.00	4.22	20.00	0.00	56.67	26.67	20.00
(ZS) + Detailed + CR-All	Direct	0.00	2.33	0.00	16.67	66.67	36.67	6.67
	Post-FDG	3.33	7.00	16.67	0.00	63.33	26.67	23.33
GPT-4o								
ZS	Direct	13.33	19.30	0.00	0.00	40.00	56.66	6.67
	Post-FDG	23.33	36.02	0.00	0.00	60.00	13.33	13.33
ZS + CR-SYN	Direct	10.00	17.14	0.00	0.00	26.67	66.67	6.67
	Post-FDG	26.67	39.11	0.00	0.00	50.00	20.00	16.67
ZS + CR-UDF	Direct	10.00	18.54	0.00	10.00	33.33	46.67	16.67
	Post-FDG	23.33	36.52	0.00	0.00	46.67	23.33	16.67
ZS + CR-TUF	Direct	6.67	14.05	0.00	3.33	23.33	63.33	10.00
	Post-FDG	23.33	35.03	0.00	0.00	56.67	13.33	10.00
(ZS)	Direct	16.67	23.28	0.00	0.00	36.67	53.33	6.67
	Post-FDG	30.00	40.83	0.00	0.00	56.67	10.00	10.00
(ZS) + Binary	Direct	16.67	24.30	0.00	0.00	33.33	53.33	6.67
	Post-FDG	26.67	41.02	0.00	0.00	60.00	10.00	6.67
(ZS) + Detailed	Direct	16.67	28.91	0.00	0.00	36.67	43.33	16.67
	Post-FDG	30.00	44.15	0.00	0.00	56.67	13.33	3.33
(ZS) + CR-SYN	Direct	20.00	24.12	0.00	0.00	36.67	53.33	3.33
	Post-FDG	30.00	40.83	0.00	0.00	60.00	10.00	6.67
(ZS) + CR-UDF	Direct	20.00	24.12	0.00	0.00	30.00	56.67	6.67
	Post-FDG	30.00	40.83	0.00	0.00	56.67	10.00	10.00
(ZS) + CR-TUF	Direct	16.67	23.07	0.00	0.00	33.33	53.33	10.00
	Post-FDG	26.67	37.47	0.00	0.00	60.00	13.33	6.67
(ZS) + Detailed + CR-SYN	Direct	23.33	29.74	0.00	0.00	30.00	50.00	10.00
	Post-FDG	30.00	43.12	0.00	0.00	53.33	16.67	3.33
(ZS) + Detailed + CR-UDF	Direct	26.67	34.18	0.00	0.00	33.33	43.33	10.00
	Post-FDG	30.00	44.23	0.00	0.00	53.33	13.33	6.67
(ZS) + Detailed + CR-TUF	Direct	13.33	25.41	0.00	0.00	33.33	46.67	16.67
	Post-FDG	30.00	43.98	0.00	0.00	56.67	13.33	3.33
(ZS) + Detailed + CR-All	Direct	13.33	24.54	0.00	0.00	33.33	50.00	16.67
	Post-FDG	33.33	46.45	0.00	0.00	50.00	13.33	6.67

Table 10: Error results on Def_ArXiv set.

Prompt Strategy	Preamble	Pass↑	FEO↑	TRO↓	IVI↓	SYN↓	UDF↓	TUF↓
miniF2F-Test								
ZS	Direct	25.41	48.90	1.23	1.23	6.15	23.77	7.38
	Post-FDG	67.21	81.88	0.00	0.00	3.28	2.87	5.33
(ZS) + Detailed	Direct	37.30	63.28	2.05	1.23	5.74	9.02	8.61
	Post-FDG	83.61	91.47	0.00	0.00	2.05	0.82	3.28
Def_Wiki-Test								
ZS	Direct	10.87	16.43	8.70	8.70	15.22	52.17	13.04
	Post-FDG	34.78	43.19	6.52	0.00	23.91	19.57	28.26
(ZS) + Detailed	Direct	19.57	23.77	8.70	8.70	8.70	47.83	10.87
	Post-FDG	43.48	50.76	6.52	0.00	17.39	10.87	28.26
Def_ArXiv								
ZS	Direct	13.33	19.30	0.00	0.00	23.33	66.67	6.67
	Post-FDG	23.33	36.02	0.00	0.00	60.00	13.33	13.33
(ZS) + Detailed	Direct	16.67	28.91	0.00	0.00	23.33	46.67	20.00
	Post-FDG	30.00	44.15	0.00	0.00	56.67	13.33	3.33

Table 11: Symbolic refinement of GPT-4o results on three dataset.

Prompt Strategy	Preamble	Pass↑	FEO↑	TRO↓	IVI↓	SYN↓	UDF↓	TUF↓
GPT-4o								
Soft-IFDC	Direct	6.52	11.45	8.70	0.00	17.39	71.74	2.17
	Post-FDG	19.57	29.65	0.00	0.00	34.78	30.43	26.09
Hard-IFDC	Direct	4.35	11.86	10.87	0.00	10.87	69.57	6.52
	Post-FDG	19.57	26.95	0.00	0.00	36.96	21.74	39.13
(ZS) + Soft-IFDC + Binary	Direct	15.22	20.47	8.70	2.17	15.22	58.70	10.87
	Post-FDG	41.30	51.09	6.52	0.00	26.09	10.87	26.09
(ZS) + Soft-IFDC + Detailed	Direct	15.22	20.20	8.70	2.17	13.04	56.52	13.04
	Post-FDG	41.30	51.26	6.52	0.0	23.91	10.87	26.09

Table 12: Prompt-FDG results on Def_Wiki test set.