KELPS: A Framework for Verified Multi-Language Autoformalization via Semantic-Syntactic Alignment

Jiyao Zhang [1 2] **Chengli Zhong** [1 2] **Hui Xu** [1] **Qige Li** [1] **Yi Zhou** [1 2]

Abstract

Modern large language models (LLMs) show promising progress in formalizing informal mathematics into machine-verifiable theorems. However, these methods still face bottlenecks due to the limited quantity and quality of multilingual parallel corpora. In this paper, we propose KELPS (**K**nowledge-**E**quation based **L**ogical **P**rocessing **S**ystem), a neuro-symbolic framework for synthesizing multiple high-quality formal languages (Lean, Coq, and Isabelle) from informal mathematical text. First, we translate natural language into Knowledge Equations (**KEs**), a novel language that we designed, theoretically grounded in assertional logic. Next, we convert them to target languages through rigorously defined rules that preserve both syntactic structure and semantic meaning. This process yielded a parallel corpus of over 60,000 problems. Our KELPS translator, fine-tuned on this dataset, finally achieves a 96.2% syntactic accuracy (pass@1) on MiniF2F with one-time automated grammar correction, outperforming SOTA models such as Deepseek-V3 (87.8%) and Herald (90.3%) across multiple datasets.

1. Introduction

Formalizing mathematical semantics as machine-verifiable codes has been a fundamental pursuit since Leibniz to Wu Wen-Tsun (Wu, 2001), as informal statements' ambiguity impedes proof verification, particularly in advanced mathematics. Many sophisticated mathematical proofs span hundreds of pages and require extensive verification by experts, while such verification cannot eliminate minor errors or critical flaws.

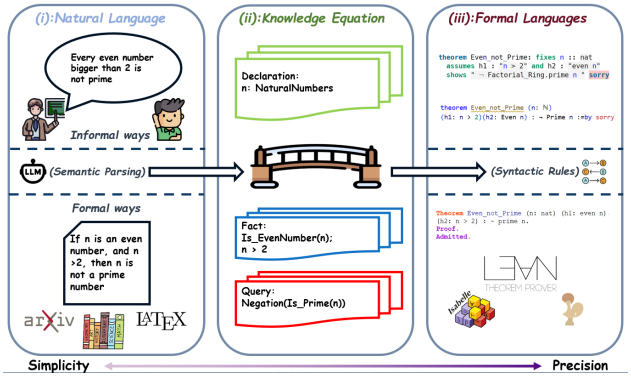

Figure 1. An overview of the relationships between natural language (NL), Knowledge Equations (KEs), and formal languages (FLs). KEs are extracted from NL through semantic parsing, then transformed into various FLs via specific syntactic rules.

A rigorous formal system is therefore essential for unambiguous mathematical representation. To bridge the gap between informal mathematics and such systems,modern proof assistants (e.g., Coq, Lean) have been developed, demonstrating significant potential through landmark achievements such as the formalization of the Four-Color Theorem in Coq (Gonthier et al., 2008) and the Liquid Tensor Experiment (Scholze, 2022) in Lean4. However, manually writing formal proofs remains laborious, creating a bottleneck for widespread adoption. Automating this process is thus critical, and we focus specifically on statement autoformalization—a necessary prerequisite for full proof automation.

The autoformalization challenge has been approached through evolving methodologies. Initial efforts (Wang et al., 2018) framed this as a machine translation task, employing neural models to convert LaTeX-written text into Mizar formal language. While subsequent work leveraging large-scale NL-FL parallel datasets through fine-tuning showed promising results, these approaches face inherent limitations due to data scarcity and lack of diversity in available NL-FL pairs. Recent advances (Wu et al., 2022) have shifted toward exploiting LLMs' in-context learning capabilities, with complementary strategies emerging, including forward

*Equal contribution [1]School of Information Science and Technology, University of Science and Technology of China, Hefei, China [2]USTC Knowledge Computing Lab. Correspondence to: Jiyao Zhang <ambitious_777@mail.ustc.edu.cn>, Yi Zhou <yi_zhou@ustc.edu.cn>.

The second AI for MATH Workshop at the 42^{nd} International Conference on Machine Learning, Vancouver, Canada.

translation of natural language problems (Ying et al., 2024) and back-translation of formal corpora (Gao et al., 2025). However, persistent challenges remain in ensuring high-quality synthetic data generation and maintaining rigorous semantic alignment between natural and formal language representations.

To address this problem, We introduce KELPS, a rule-based framework designed to synthesis multiple formal statements, including Lean4 (Moura & Ullrich, 2021), Coq (Barras et al., 1999), and Isabelle (Paulson, 1994). As illustrated in Figure 2, KELPS consists of three core components.

(1) Semantic Parsing. The input natural language is first translated into an intermediate formal representation—the *Knowledge Equation*. The formal definition and implementation details are in Section 3.1.

(2) Syntactic Alignment. The results in stage 1 are then parsed and converted into various formal languages via parallel complex rules. Compilers will help to check its correctness.

(3) Semantic Verification. Formal statements generated in Stage 2, even though compiled successfully, still need semantic checks to be correct. We adopt the approach proposed in (Xin et al., 2025), leveraging a LLM-as-a-Judge framework.

We propose a data augmentation and expert iteration framework for progressive model enhancement. Our approach combines two key strategies: (1) A data synthesis module that efficiently generates various formalization data through randomized combinations of concepts and operators; (2) An expert iteration process where our model successfully parsed >60,000 formalized problems. The synthesized dataset was used to train our KELPS translator model. Comprehensive evaluations demonstrate its superior performance, achieving a raw **88.9%** (pass@1) syntactic accuracy in the MiniF2F test, outperforming baseline models including Herald (81.3%), DeepSeek-v3 (81%) and Llama (61.4%).

Our main contributions are as follows.

- We develop the first unified framework for automatically synthesizing multiple parallel NL-FL data via an intermediate representation, enabling quality-controlled data-generation process.
- We introduce a 60K dataset covering K-12 to undergraduate mathematics, combining real-world problems and synthetic examples with verified formalizations.
- We introduce the KELPS translator, which achieves 88.9% accuracy on MiniF2F (+7.6% over Herald), shows substantial improvements over existing baselines on mainstream evaluation benchmarks.

2. Related Work

2.1. Formal System

Natural languages exhibit inherent ambiguity in both textual and symbolic forms (Ganesalingam, 2013), complicating syntactic analysis and semantic extraction. This has motivated systematic efforts to formalize mathematical expressions through logical frameworks (Trybulec, 1989; Gordon, 2000; Carneiro, 2024). Existing approaches fall into two main distinct categories: controlled natural languages and formal language systems.

Controlled Natural Language (CNL) systems employ restricted natural language subsets with predefined grammars, as exemplified by Mizar (Grabowski et al., 2010) and MathNat (Humayoun & Raffalli, 2010). Recent advances include grammatical frameworks like GF (Ranta, 2004), with GFLean (Pathak, 2024) demonstrating direct text-to-Lean parsing.

Formal language systems (Lean, Coq) uniformly represent theorems through three core elements (**Declaration**, **Fact**, **Query**) despite differing in styles. To our knowledge, this work presents the first automated framework supporting multi-formal-language translation. Compared to existing formal languages, our knowledge equation framework achieves superior expressiveness and natural language alignment while maintaining simpler structure and better extensibility.

2.2. Autoformalization

Autoformalization constitutes a specialized machine translation task that transforms natural language statements into formal representations while preserving semantic content and complying with target syntax requirements. Initial investigations explored neural approaches like (Wang et al., 2018; Cunningham et al., 2023), demonstrating the feasibility of this paradigm.

Current research on LLM-based autoformalization primarily follows two dominant approaches: (1) few-shot in-context learning (Wu et al., 2022; Patel et al., 2023; Zhou et al., 2024), and (2) fine-tuning LLMs on NL-FL pairs (Lu et al., 2024a;b; Gao et al., 2025). While the latter has shown promising results with **96**% (pass@128) accuracy in MiniF2F (Zheng et al., 2021), performance drops to just **16**% (pass@128) on the College CoT benchmark, revealing the critical limitation of NL-FL data scarcity.

A parallel research direction addresses the more challenging task of formal proof generation, where natural language proofs often diverge substantially from their formal counterparts. Current approaches include: (1) proof decomposition into draft skeletons with subsequent completion (Jiang et al., 2022; Wang et al., 2023), and (2) direct neural translation of

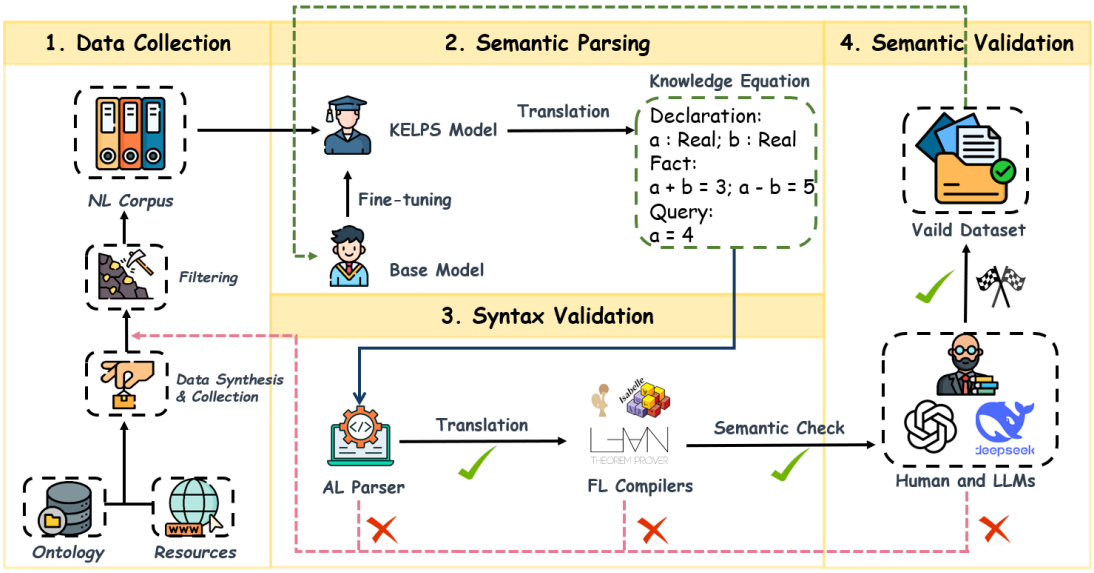

Figure 2. Overview of Data Synthesis Process. **(a) Data Collection**. We gather problems from various sources, including online resources and exercise sets, and construct an ontology library of relevant concepts and theorems. Through filtering and data synthesis strategies, we obtain a natural language (NL) corpus. **(b) Semantic Parsing**. We employ the KELPS model to perform semantic parsing, translating natural language problems into knowledge equations. The initial iteration of data is obtained through annotation. **(c) Syntax Validation**. The knowledge equations generated in **(b)** are validated by the AL Parser. Problems that pass validation are then converted into other formal languages via rule-based transformation. **(d) Semantic Validation**. Data that passes the compiler validation in the previous stage undergoes semantic review by both LLMs and human experts. Finally, the verified data is incorporated into the dataset, which is then used to continuously train the baseline model.

informal proofs (Wang et al., 2024; Shao et al., 2024). The first approach uses language models to complete proof steps within a structured framework, whereas the second aims for complete automated translation.

2.3. NL-FL Dataset Generation

The scarcity of high-quality, large-scale natural language-formal language (NL-FL) parallel datasets remains a fundamental challenge in autoformalization research. Since hiring domain experts for annotation is expensive and inefficient, recent work has investigated leveraging large language models for scalable dataset generation. Existing approaches include two main categories:

The first line of work (Jiang et al., 2023; Li et al., 2024; Ying et al., 2024) exploits the wide availability of informal mathematical texts, using LLMs to translate NL statements into FL statements. Although this pipeline has yielded some large-scale datasets, it remains hampered by several limitations: the translation pipeline requires extensive post-processing, often produces low-quality statements, and faces challenges due to the scarcity of cutting-edge domain data.

The alternative paradigm (Wu et al., 2024; Gao et al., 2025) initiates from formal language corpora, employing LLMs for backward translation to natural language. While valuable, this approach remains fundamentally constrained by the scope and completeness of existing formal libraries.

Building upon existing work (Huang et al., 2024; Liu et al., 2025) that generates diverse problems through randomized concept selection from a predefined concept library, we introduce a more controlled synthesis approach. Our method employs structured **Concept-Operator** templates to achieve three key advances: **(1)** ensuring comprehensive theorem coverage through a Concept-Operator library, **(2)** enabling easy-to-hard theorem synthesis via predefined templates, and **(3)** generating theorem diversity through various Concept-Operator combinations.

3. Methodology

In this section, we present our core methodologies for statement autoformalization and dataset construction. Section 3.1 establishes our theoretical foundations (**Assertional Logic** and **Knowledge Equations**). Section 3.2 presents the complete system architecture, and Section 3.3 details our synthetic data generation strategy.

3.1. Assertional Logic and Knowledge Equation

We introduce Assertional Logic (AL), a knowledge representation system with formally specified syntax.

3.1.1. An Introduction to Assertional Logic

Assertional Logic (AL) (Zhou, 2017), is an extension of first-order logic with enhanced power as expressive as higher-order logic, while these representations are usually human-friendly.

Definition 3.1. The **Syntactic structure** of a given domain in AL is a tripe $\langle \mathcal{I}, \mathcal{C}, \mathcal{O} \rangle$. $\mathcal{I}$ is the collection of individuals, corresponding to objects in the domain. $\mathcal{C}$ is the collection of concepts, representing all sets of objects that have some properties in common. $\mathcal{O}$ is the collection of all operators, which acts among concepts and individuals like a function.

This structure has a natural correspondence with set theory, where individuals map to elements, concepts to sets, and operators to functions.

Definition 3.2. An **assertion** is the of form

$$a = b \tag{1}$$

where a and b are two terms. From a semantic perspective, it claims that the left and the right side refer to the same thing. A $term$ is an individual, either an atomic individual $a \in \mathcal{I}$ or the compound individuals $O(a_1, ..., a_n)$. Where O represents an operator on some individuals $a_1, ..., a_n$.

Building upon AL, we presented Knowledge Equations (KE) to represent all knowledge with the same form. There is an example in Fig 1.

3.1.2. Translate KE into multiple language

One of the biggest benefits of AL is that it could uniformly formalize every knowledge into assertions of the form $a = b$. Consequently, all mathematical assertions can be systematically translated into other formal languages, including but not limited to Lean and Coq.

Unlike GFLean (Pathak, 2024) which handles natural language in its entirety, our method specifically targets assertions, concepts, and operators. The simplified syntax works with just a handful of core elements, enabling efficient translation to various formal languages.

In summary, a mathematical question can be divide into three parts: **Declaration**, **Fact** and **Query**. And we give each of their formal definitions.

Definition 3.3. Declaration part of a knowledge equation has the form of

$$\text{var} : \text{ConceptType} \tag{2}$$

where "var" represents a free variable (belongs to Individual in AL) and "ConceptType" is a defined concept. Its semantic meaning is similar to "Assume x is an integer ..." and the syntactic part of "fixes n :: int" in Isabelle or "$(n : \mathbb{Z})$" in Lean4.

Definition 3.4. Fact part of a knowledge equation has the form of

$$[\text{Assertion1}; \text{Assertion2}; \text{Assertion3}...] \tag{3}$$

During the translation process, KELPS model will systematically capture all known information, (including inferable propositions such as "Set S is finite"), and formalize them as assertions. By leveraging the equivalence of a proposition $A \equiv (A = \text{True})$, this representation ensures seamless translation to various formal languages.

Definition 3.5. Query part shares the syntactic structure with facts, but differs in their semantic meaning.

$$\text{Assertion} \tag{4}$$

The assertions in the Query typically represent propositions requiring proof (for closed-form problems) or propositions we aim to investigate (for open-ended questions). For the later situation, we use " ? " as syntax sugar to replace the real item. This is just like " sorry " in Lean 4.

We designed the Backus-Naur Form (BNF) of Knowledge Equations based on the formal definitions above. And by using ANTLR4, we implemented an extensible parser framework that automatically transforms Knowledge Equations into equivalent representations in target formal languages. The correctness is guaranteed by translation rules designed by human experts:

$$C_{\text{KE}} \mapsto C_{\text{TL}}, \quad O_{\text{KE}} \mapsto O_{\text{TL}} \tag{5}$$

where C_{KE} and O_{KE} represents concepts and operators in KE, C_{TL} and O_{TL} represents the same semantic object in the target language.

3.2. KELPS Framework

In this subsection, we present the KELPS framework (illustrated in Figure 2), which comprises three core components: Semantic Parsing, Syntactic Validation, and Semantic Validation. We will now describe each component in sequence.

3.2.1. Semantic Parsing

Large-scale data annotation remains highly laborious. We develop an iterative pipeline that first fine-tunes DeepSeek-Math-7B-Base on **1,200** manually annotated examples (Fig. 2), then automatically processes unannotated data. The model's outputs undergo syntactic and semantic checks, with validated results expanding the training set through multiple refinement cycles. After seven iterations, our model processed $>$ 50K validated samples.

Table 1. Comparison between KELPS Dataset and Existing Benchmarks (Dataset Scale, Language Support, and Knowledge Coverage)

Benchmark	Dataset Size	Language	Multi-Supported	Coverage
MiniF2F	244	Lean4, Isabelle	✓	High-School
PutnamBench	1,709	**Lean4, Isabelle, Coq**	✓	High-School & Undergraduate
FormalMATH	5,560	Lean4	×	High-School & Undergraduate
Lean Workbook	57k	Lean4	×	High-School
KELPS Dataset	**60k**	**Lean4, Isabelle, Coq**	✓	High-School & Undergraduate

3.2.2. Syntax Validation

To ensure the syntactic correctness of the semantic parsing results obtained from Section 3.2.1, we implement a two-stage validation process.

We first perform grammar verification using our ANTLR4-based knowledge equation parser that guarantees strict compliance with formal specifications. Subsequently, we employ the target language's compiler to verify its final correctness.

For statements processed by the knowledge equation parser, approximately 80-90% successfully pass validation through the target language compiler typically. The remaining cases primarily involve minor type conversion errors, which we analyze comprehensively in Appendix C.

3.2.3. Semantic Validation

To ensure the semantic correctness of the semantic parsing results obtained from Section 3.2.2, our evaluation framework incorporates insights from mainstream methods (Wu et al., 2022; Gao et al., 2025). However, we observe that back-translation from formal to natural language fails to preserve semantic fidelity, thereby introducing extra measurement errors. Furthermore, the binary **True/False** classification criterion is insufficient for the precise measurement of semantic alignment in formalized expressions.

We evaluate formalized statements through a graded alignment assessment framework **(0-5 scale)** with corresponding natural language expressions. Uncompilable statements are automatically assigned a score of 0. The evaluation is performed using DeepSeek-V3 (version 250324) and Claude (version sonnet-4-20250514-thinking) with default parameters.

In designing the scoring mechanism, we observe that DeepSeek-V3 consistently assigned higher scores than Claude, with an average difference of about 1 point. This stems from differing evaluation criteria: DeepSeek-V3 is more lenient, often giving full marks for generally reasonable outputs, whereas Claude is more stringent, penalizing even minor errors. To reduce the impact of such bias, we manually analyzed sampled results and confirmed the consistency of this pattern (see Appendix C). Therefore, we adopt the arithmetic mean of the two scores as the final metric to better reflect the combined evaluation perspectives.

3.3. AL-FL Data Generation

This section presents the KELPS dataset construction methodology - a corpus of 60,000+ NL-FL and AL-FL pairs supporting major formal languages.

Our pipeline first establishes a mathematics ontology, then employs dual generation strategies: (1) translation of natural language problems into AL representations, and (2) template-based synthesis of NL-AL pairs through a combination of templates. Implementation details follow in subsequent sections.

3.3.1. Building the Mathematical Ontology

We manually constructed a mathematics ontology covering most K12 and selected undergraduate-level mathematical topics, which comprises **6** major topics, **40** core concepts, and **180** operators. This ontology development effort required approximately three weeks of work by two mathematical graduate students.

The complete specifications and structural details of the ontology are provided in Appendix A.

3.3.2. Collection and Processing Data

This module discusses two main steps in building our dataset from NL problems —- problem collection, filtering, and translation.

NL Problems Collection

The Numina (Li et al., 2024) dataset represents the largest and most comprehensive open-source collection of K12 mathematics materials, incorporating diverse sources ranging from AIME competition problems to Chinese K12 curriculum content. In this study, we selected Numina as our primary reference dataset due to its unique coverage that subsumes content from numerous other mathematical datasets.

Filtering and Translation

We first filtered out unsupported problems from the Numina dataset, including out-of-ontology mathematical ques-

tions, retaining approximately 100,000 items. Subsequently, we removed problems unsuitable for formalization, such as questions with graphical representations. Finally, we yielded a corpus of 70,000 problems for translation.

We adopted an expert-iterative approach for continuous problem parsing and successfully translated **50,000** problems after seven iterations.

3.3.3. Synthesis Strategies

This subsection presents our synthetic data generation framework, which consists of two key components: template creation based on our mathematical ontology, and a systematic template combination strategy.

Templates Creation

Using the strong in-context learning capabilities of LLM, we found that providing just 3-shot examples was sufficient for the model to correctly learn patterns of target concepts and operators. For our template-based generation tasks, DeepSeek-v3 (Liu et al., 2024) achieved a 90% syntactic accuracy.

In addition to templates derived from individual concepts, we extract specialized templates from concrete problem instances. For example, the problem "Find all integers such that [a is a prime and $a < 15$]" is abstracted into a template where the condition "[a is a prime and $a < 15$]" is replaced with a generic "[property]" placeholder. This approach enhances the model's capacity to generate diverse problem variations.

Templates Combination

However, this approach tended to produce problems with limited diversity, and the model occasionally made errors due to incomplete conceptual understanding. To address these limitations, we developed a composite template strategy that systematically combines templates of varying complexity - from basic concept applications to advanced problem types. We ultimately constructed a corpus of 50+ high-quality templates. The combination of templates enables the model to go beyond simple problem generation, demonstrating remarkable creativity.

The complete set of prompting templates employed to generate synthetic data is provided in Appendix B.2.

4. Experiments

We conduct a comprehensive series of experiments to evaluate both the performance of the KELPS translator and the quality of the KELPS dataset. Section 4.1 details the experimental setup and configurations. Section 4.2 presents the main results of multiple benchmarks. Section 4.3 provides an ablation study to analyze the effect of different components. Finally, Section 4.4 offers a further analysis of the influencing factors behind the results.

4.1. Experimental Setup

Fine-tuning. We employ DeepSeek-Math-7B-Base as our base model and conduct supervised fine-tuning on the KELPS dataset using a full-parameter training approach.

Dataset. To evaluate the performance of our system, we conduct comparative benchmarks across three mathematical datasets: MiniF2F, FormalMATH, and Numina-Hard. These established benchmarks comprehensively cover olympiad/undergraduate mathematics through diverse problem types, ideal for formalization testing.

- **MiniF2F**. (Zheng et al., 2021) A widely adopted multilingual benchmark for auto-formalization tasks, comprising K12-level mathematical problems. In this work, we only evaluate its test set.
- **FormalMATH**. (Yu et al., 2025) A formalized mathematical benchmark that spans Olympic competitions and undergraduate-level problems in multiple domains. We randomly select 200 problems across various mathematical domains for evaluation.
- **Numina-Hard**. The Numina-Hard dataset comprises 300 challenging problems randomly selected from the KELPS dataset.

Due to the comprehensiveness and practicality of **Mathlib** (Blokpoel, 2024), we adopt Lean4 as our primary formalization language. All experiments are conducted in Lean4 by default. In addition, we evaluate the **NL** to **Isabelle** translation on the MiniF2F benchmark. Herald's results (Gao et al., 2025) were excluded as their experimental setup diverged from our task requirements.

Experimental Process. To validate our model's capabilities, we employ the experimental pipeline illustrated in Fig 2. It comprises three core steps: **semantic parsing**, **syntax validation**, and **semantic validation**, with detailed descriptions provided in Section 3.3. For the Herald model, we rigorously maintain the original configuration reported in (Gao et al., 2025). The implementation details and hyperparameter settings for all other models are provided in Appendix B.1.

All KELPS experiments used NL-Lean4 fine-tuning and translated directly into Lean4 in this chapter, except for **MiniF2F-Isabelle**, which employed NL-AL training followed by rule-based Isabelle translation.

Our experimental environment utilizes Lean v4.19.0 (with Mathlib4 of the same version), and Isabelle 2025 (March

Table 2. Evaluating Performance Across Different Models and Datasets. We highlight the top-performing results for syntactic accuracy in red and those for semantic accuracy in green. Syntactic accuracy is determined by whether the code passes compiler verification, while semantic accuracy scores are obtained through LLM majority voting. The version of Deepseek-V3 used is **DeepSeek-V3-0324**. Given the substantial volume of problems, we selectively sampled **200** questions from FormalMATH and **300** from Numina-Hard for experiments. CR (Corrected Results) denotes the outcomes after rectifying erroneous problems by incorporating compiler error messages.

Model	MiniF2F		MiniF2F-Isabelle		Numina-Hard		FormalMATH	
	Syntax	Semantic	Syntax	Semantic	Syntax	Semantic	Syntax	Semantic
DeepSeek-V3 (671B)	80.7%	**3.90**	56.5%	2.67	79.3%	3.73	58.3%	2.84
Herald (7B)	81.3%	3.39	–	–	83.7%	2.85	73.8%	2.84
LlaMa-3 (8B)	59.0%	2.28	54.3%	2.04	54.1%	1.94	37.7%	1.37
KELPS (7B)	**88.9%**	3.83	**82.2%**	**3.22**	**94.3%**	**4.29**	**74.3%**	**2.99**
DeepSeek-V3+CR	87.8%↑36.7	3.92	–	–	88.1%↑42.4	3.93	74.8%↑39.6	3.63
Herald+CR	90.3%↑58.4	3.75	–	–	90.5%↑54.1	3.12	84.9%↑47.1	3.28
LlaMa-3+CR	85.4%↑64.4	2.82	–	–	84.8%↑66.7	2.62	71.2% ↑53.8	2.71
KELPS+CR (7B)	**96.2%**↑65.9	**4.13**	–	–	**98.4%**↑72.1	**4.47**	**91.1%**↑65.4	**3.69**

2025). The header files and executable code used in our experiments are included in our supplementary materials.

4.2. Main Results

In this section, we present a systematic comparison between the KELPS model and other baseline models across various benchmark datasets. Our evaluation framework assesses two critical dimensions of model performance: (1) **syntactic accuracy**, measuring formal correctness, and (2) **semantic accuracy**, evaluating meaningful correspondence to mathematical truth. We note that Herald's calculation method for the pass rate differs from ours: we evaluate **syntactic and semantic correctness separately**, whereas they consider a case as passed **only when both syntax and semantics are correct**.

Syntactic Accuracy quantifies the formal correctness of model outputs by measuring their ability to pass automated compiler verification. The overall pass rate is equal to the ratio of **compiler-valid statements** to **all problems**.

Semantic Accuracy assesses the mathematical equivalence between formally verified statements and their original natural language formulations, following the evaluation protocol established in 3.2.3. The semantic scores reported in Table 2 represent the **average performance** across all problems. Problems that fail to compile are assigned a score of **zero**.

As summarized in Table 2, our experimental results demonstrate that the KELPS translator maintains robust performance across all evaluation metrics and datasets. Notably, while other models exhibit significant accuracy gaps between Lean and Isabelle formalizations, the KELPS translator's consistent performance confirms its capability for cross-formal-language translation.

The results demonstrate that the KELPS translator excels at formalizing natural language sentences into various formal representations. A case study of the formalization results of the KELPS translator is shown in Appendix C.

4.3. Ablation Study

In this subsection, we perform ablation studies to evaluate the effectiveness of both the synthetic data strategy and the KE representation.

Effectiveness of KE. We observe that problems with syntactic errors exhibit varying error types and correction difficulties. To evaluate the effectiveness of the KE representation, we employ DeepSeek-v3 to correct these syntactically flawed problems, using the following inputs: (1) the original problem statement, (2) raw Lean4 code, and (3) compiler error messages. As shown in Table 2, models trained with KE-generated problems demonstrate significantly higher correction success rates.

The error correction pass rate is also influenced by the correction model's capability. Notably, even for relatively more challenging problems (where the original pass rate was already high), KELPS maintains significantly superior correction success rates on the remaining unsolved problems. For further discussion, see Section 4.4. The complete prompts and experimental configurations are provided in the appendix B.2.

Effectiveness of Synthetic Data. We compare the results under different configurations. All models were trained using a subset of the KELPS dataset containing approximately **15,000** Numina-Basic NL-AL pairs parsed from Numina combined with various categories of synthetic data. To systematically evaluate the effects, we controlled two key factors: (1) dataset diversity and quality by varying the

Table 3. We compare the performance of KELPS trained on datasets of varying scales and composition ratios across different benchmark datasets, where the 'Dataset ratio' denotes the proportion of **informal data** from natural language problems to model-generated **synthetic data** in the training corpus.

Dataset Ratio	Dataset Size	MiniF2F		Numina-Hard		FormalMATH	
		Syntax	Semantic	Syntax	Semantic	Syntax	Semantic
1 : 0	14k	81.6%	3.75	91.7%	4.33	60.1%	2.54
1 : 0.5	21k	87.6%	4.08	**94.5%**	**4.51**	70.3%	3.14
1 : 1	28k	88.4%	**4.08**	93.4%	4.49	70.9%	3.25
1 : 1.5	35k	**88.9%**	4.05	94.3%	4.49	**74.3%**	**3.29**

templates used for synthetic data generation, (2) the mixing ratio between synthetic and authentic data samples.

As shown in Table 3, the experimental results demonstrate that our synthetic data augmentation approach significantly improves the performance of the model. Compared to the limited data variety in Numina, synthetic data significantly enriches training diversity, yielding notable performance gains by incorporating only 7,000 additional problems. We also observe a non-linear relationship between the training data scale and the performance of the model. When the dataset is small ($<$ 20K samples), increasing its size significantly improves both semantic understanding and grammatical accuracy. However, beyond a certain threshold, further scaling yields diminishing returns. This implies that prioritizing **high-quality, diverse training data** may be more effective than simply pursuing larger quantities.

4.4. Analysis

In this subsection, we provide an in-depth analysis of the experimental outcomes and identify key factors that influence model performance. These factors represent the primary targets for improvement in our subsequent research.

Effectiveness of KE Framework: We will discuss how the KE's data augmentation pipeline enhances experimental performance. The task of autoformalization can be regarded as an alignment between NL and FL.

Corollary 4.1. *This alignment can be further categorized along the following two dimensions:*

$$AG(NL, FL) = AG_{syn}(NL, FL) + AG_{sem}(NL, FL)$$

where AG denotes the overall alignment gap between the informal statement (NL) and the formal statement (FL). AG_{syn} represents the *syntactic alignment gap*, while AG_{sem} denotes the *semantic alignment gap*.

Corollary 4.2. *The term AG_{syn} primarily depends on the syntactic rules and stylistic conventions of the FL. And AG_{syn} coule be decomposed into two subcomponents*

$$AG_{syn}(NL, FL) = AG_{syn}(NL, AL) + AG_{syn}(AL, FL)$$

(1) The KE framework's primary contribution is its structured data generation process (**NL→AL→Lean4**), which ensures high-quality training data through constrained syntactic-semantic alignment. During the construction of our dataset, natural language is **first translated into a strictly constrained format (on AL)**, which is then **directly translated into Lean4 using predefined transformation rules**. This process effectively unifies diverse syntactic rules within KE's grammatical framework, ensuring the consistency of the obtained Lean4 data format and significantly accelerating both the model's learning speed and performance(Table 2).

(2) Another additional benefit of KE is its ease of correction, as mentioned in the formula above. When we consider syntactic alignment in terms of $\text{AG}_{\text{syn}}(\text{NL}, \text{AL})$ and $\text{AG}_{\text{syn}}(\text{AL}, \text{FL})$, models trained under the KE framework ensure alignment for $\text{AG}_{\text{syn}}(\text{NL}, \text{AL})$, while $\text{AG}_{\text{syn}}(\text{AL}, \text{FL})$ alignment requires more detailed Lean4 knowledge, such as dependent types. As shown in Table 2, our KELPS model achieves a significantly higher error correction success rate (**averaging 60%**) compared to other models like DeepSeek-v3 (**averaging 40%**), demonstrating that the KE framework effectively reduces downstream post-processing costs.

Quality of Alignment between NL-FL pairs: Since the semantic verification module that relies on a large language model as a referee is not completely reliable, instances of misalignment between NL and FL persist in the training dataset. Common misalignment patterns include misinterpretation of problem semantics and omission of problem-specific constraints.

Our preliminary experiments revealed that semantically erroneous data can mislead models to generate incorrect responses. To address this issue, We employed a rigorous manual verification process to curate a high-quality subset of 15,000 precisely aligned NL-FL pairs for experiments.

The Diversity and Coverage of the KELPS Ontology: Due to limited time and resources, we only modeled core mathematical concepts in natural language. While these atomic concepts are theoretically sufficient to represent most mathematical knowledge, our experiments revealed that the

model tends to **self-construct undefined operators** (shown in Appendix C). These operators failed syntax validation because of the absence of corresponding transformation rules.

Therefore, to enhance both the practical utility and expressive power of Knowledge Equations, our future research will focus on comprehensive semantic modeling of natural language in its entirety.

5. Discussion

Automated Construction of Advanced Ontology. To ensure knowledge accuracy, we adopt expert-guided ontology construction with LLM assistance. The challenges of this task lie in two aspects: (1) Collecting concepts existing in natural language, (2) establishing formal representations and transformation rules for these concepts. While LLMs excel at extracting natural language concepts, creating formal representations and transformation rules still requires expert validation due to LLMs' limited domain knowledge. However, in the future, a mature agent framework holds the potential to automate these tasks fully.

Extensibility Properties of KE. As demonstrated in Section 3.1, KE's theoretical foundation ensures its capability to represent content based on **Set Theory**. Consequently, KE can represent more complex mathematical definitions, such as those in abstract algebra. A further question arises: **Can KE represent knowledge from other disciplines, such as physics or chemistry**? While existing research has confirmed that formal methods can encode theories in these domains ((Tooby-Smith, 2025; Bobbin et al., 2024)), experimental sciences often deal with phenomena lacking strict theoretical explanations. The most promising candidates for KE extension are established theoretical frameworks (like theoretical mechanics), which we plan to explore in the future.

Limitations. The current KELPS framework has two major limitations: **(1)** Inability to represent mathematical proofs. Modern proof assistants rely heavily on tactics with well-defined semantics for proof simplification. While KELPS's core methodology could theoretically be extended to proof translation, this capability remains unimplemented. **(2)** Absence of dependent types. By grounding its theoretical foundation in set theory rather than type theory, KELPS gains usability at the cost of representational precision. This design choice inherently limits its capacity to express certain mathematical constructs with type-theoretic dependencies.

6. Conclusion

In this work, we propose KELPS, a novel three-stage framework for autoformalization. Our method introduces an intermediate representation—Knowledge Equation, which translates natural language into multiple formal languages. This representation aligns more closely with natural language, thus improving model accuracy, while the expert-crafted transformation rules guarantee syntactic correctness.

We introduce a large-scale parallel dataset comprising over 60,000 NL-FL pairs spanning distinct mathematical subfields, with multi-language support (Lean, Coq, Isabelle). Additionally, we propose an LLM-based synthetic data generation strategy that controls difficulty levels and targets specific concepts/operators, effectively enhancing data diversity. Fine-tuning the KELPS translator on this dataset achieved 88.9% syntactic accuracy on the MiniF2F dataset, outperforming SOTA models like Deepseek-V3 81% and Herald 81.3% on MiniF2F.

In summary, this work makes three contributions: the design of Knowledge Equations as a novel formal language, including its complete BNF specification and parser implementation; the release of KELPS Dataset, a large-scale multilingual formal language dataset; and the development of a high-performance model that achieves new SOTA accuracy. Due to their simplified formalism, Knowledge Equations demonstrate strong potential as both an educational and research-oriented formal language, despite limitations in automated type conversion between multiple concepts.

In the future, we plan to extend and refine this methodology along two key dimensions: expanding Knowledge Equations' coverage to advanced mathematical domains, and strengthening its theoretical foundations to address type system conversion errors. We posit that Knowledge Equations possess the potential to emerge as a universal, machine-learnable language.

7. Impact Statement

This paper presents work whose goal is to advance the field of Machine Learning. There are many potential societal consequences of our work, none which we feel must be specifically highlighted here.

References

Barras, B., Boutin, S., Cornes, C., Courant, J., Coscoy, Y., Delahaye, D., de Rauglaudre, D., Filliâtre, J.-C., Giménez, E., Herbelin, H., et al. The coq proof assistant reference manual. *INRIA, version*, 6(11):17–21, 1999.

Blokpoel, M. mathlib: A scala package for readable, verifiable and sustainable simulations of formal theory. *Journal of Open Source Software*, 9(99):6049, 2024.

Bobbin, M. P., Sharlin, S., Feyzishendi, P., Dang, A. H., Wraback, C. M., and Josephson, T. R. Formalizing chem-

ical physics using the lean theorem prover. *Digital Discovery*, 3(2):264–280, 2024.

Carneiro, M. Lean4lean: Towards a verified typechecker for lean, in lean. *arXiv preprint arXiv:2403.14064*, 2024.

Cunningham, G., Bunescu, R. C., and Juedes, D. Towards autoformalization of mathematics and code correctness: Experiments with elementary proofs. *arXiv preprint arXiv:2301.02195*, 2023.

Ganesalingam, M. The language of mathematics. In *The Language of Mathematics: A Linguistic and Philosophical Investigation*, pp. 17–38. Springer, 2013.

Gao, G., Wang, Y., Jiang, J., Gao, Q., Qin, Z., Xu, T., and Dong, B. Herald: A natural language annotated lean 4 dataset. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=Se6MgCtRhz.

Gonthier, G. et al. Formal proof–the four-color theorem. *Notices of the AMS*, 55(11):1382–1393, 2008.

Gordon, M. From lcf to hol: a short history. In *Proof, language, and interaction*, pp. 169–186. Citeseer, 2000.

Grabowski, A., Kornilowicz, A., and Naumowicz, A. Mizar in a nutshell. *Journal of Formalized Reasoning*, 3(2): 153–245, 2010.

Huang, Y., Lin, X., Liu, Z., Cao, Q., Xin, H., Wang, H., Li, Z., Song, L., and Liang, X. Mustard: Mastering uniform synthesis of theorem and proof data. *arXiv preprint arXiv:2402.08957*, 2024.

Humayoun, M. and Raffalli, C. Mathnat-mathematical text in a controlled natural language. *Special issue: Natural Language Processing and its Applications*, 46:293–307, 2010.

Jiang, A. Q., Welleck, S., Zhou, J. P., Li, W., Liu, J., Jamnik, M., Lacroix, T., Wu, Y., and Lample, G. Draft, sketch, and prove: Guiding formal theorem provers with informal proofs. *arXiv preprint arXiv:2210.12283*, 2022.

Jiang, A. Q., Li, W., and Jamnik, M. Multilingual mathematical autoformalization. *arXiv preprint arXiv:2311.03755*, 2023.

Li, J., Beeching, E., Tunstall, L., Lipkin, B., Soletskyi, R., Huang, S., Rasul, K., Yu, L., Jiang, A. Q., Shen, Z., et al. Numinamath: The largest public dataset in ai4maths with 860k pairs of competition math problems and solutions. *Hugging Face repository*, 13:9, 2024.

Liu, A., Feng, B., Xue, B., Wang, B., Wu, B., Lu, C., Zhao, C., Deng, C., Zhang, C., Ruan, C., et al. Deepseek-v3 technical report. *arXiv preprint arXiv:2412.19437*, 2024.

Liu, X., Bao, K., Zhang, J., Liu, Y., Chen, Y., Liu, Y., Jiao, Y., and Luo, T. Atlas: Autoformalizing theorems through lifting, augmentation, and synthesis of data. *arXiv preprint arXiv:2502.05567*, 2025.

Lu, J., Wan, Y., Huang, Y., Xiong, J., Liu, Z., and Guo, Z. Formalalign: Automated alignment evaluation for autoformalization. *arXiv preprint arXiv:2410.10135*, 2024a.

Lu, J., Wan, Y., Liu, Z., Huang, Y., Xiong, J., Liu, C., Shen, J., Jin, H., Zhang, J., Wang, H., et al. Process-driven autoformalization in lean 4. *arXiv preprint arXiv:2406.01940*, 2024b.

Moura, L. d. and Ullrich, S. 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*, pp. 625–635. Springer, 2021.

Patel, N., Saha, R., and Flanigan, J. A new approach towards autoformalization. *arXiv preprint arXiv:2310.07957*, 2023.

Pathak, S. Gflean: An autoformalisation framework for lean via gf. *arXiv preprint arXiv:2404.01234*, 2024.

Paulson, L. C. *Isabelle: A generic theorem prover*. Springer, 1994.

Ranta, A. Grammatical framework. *Journal of Functional Programming*, 14(2):145–189, 2004.

Scholze, P. Liquid tensor experiment. *Experimental Mathematics*, 31(2):349–354, 2022.

Shao, Z., Wang, P., Zhu, Q., Xu, R., Song, J., Bi, X., Zhang, H., Zhang, M., Li, Y., Wu, Y., et al. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300*, 2024.

Tooby-Smith, J. Heplean: Digitalising high energy physics. *Computer Physics Communications*, 308:109457, 2025.

Trybulec, A. Tarski grothendieck set theory. *Journal of Formalized Mathematics*, 1, 1989.

Wang, H., Xin, H., Zheng, C., Li, L., Liu, Z., Cao, Q., Huang, Y., Xiong, J., Shi, H., Xie, E., et al. Lego-prover: Neural theorem proving with growing libraries. *arXiv preprint arXiv:2310.00656*, 2023.

Wang, Q., Kaliszyk, C., and Urban, J. First experiments with neural translation of informal to formal mathematics. In *Intelligent Computer Mathematics: 11th International Conference, CICM 2018, Hagenberg, Austria, August 13-17, 2018, Proceedings 11*, pp. 255–270. Springer, 2018.

Wang, R., Zhang, J., Jia, Y., Pan, R., Diao, S., Pi, R., and Zhang, T. Theoremllama: Transforming general-purpose llms into lean4 experts. *arXiv preprint arXiv:2407.03203*, 2024.

Wu, W.-T. *Mathematics mechanization: mechanical geometry theorem-proving, mechanical geometry problem-solving, and polynomial equations-solving*. Kluwer Academic Publishers, 2001.

Wu, Y., Jiang, A. Q., Li, W., Rabe, M., Staats, C., Jamnik, M., and Szegedy, C. Autoformalization with large language models. *Advances in Neural Information Processing Systems*, 35:32353–32368, 2022.

Wu, Z., Wang, J., Lin, D., and Chen, K. Lean-github: Compiling github lean repositories for a versatile lean prover. *arXiv preprint arXiv:2407.17227*, 2024.

Xin, H., Li, L., Jin, X., Fleuriot, J., and Li, W. Ape-bench i: Towards file-level automated proof engineering of formal math libraries. *arXiv preprint arXiv:2504.19110*, 2025.

Ying, H., Wu, Z., Geng, Y., Wang, J., Lin, D., and Chen, K. Lean workbook: A large-scale lean problem set formalized from natural language math problems. *arXiv preprint arXiv:2406.03847*, 2024.

Yu, Z., Peng, R., Ding, K., Li, Y., Peng, Z., Liu, M., Zhang, Y., Yuan, Z., Xin, H., Huang, W., et al. Formalmath: Benchmarking formal mathematical reasoning of large language models. *arXiv preprint arXiv:2505.02735*, 2025.

Zheng, K., Han, J. M., and Polu, S. Minif2f: a cross-system benchmark for formal olympiad-level mathematics. *arXiv preprint arXiv:2109.00110*, 2021.

Zhou, J. P., Staats, C., Li, W., Szegedy, C., Weinberger, K. Q., and Wu, Y. Don't trust: Verify–grounding llm quantitative reasoning with autoformalization. *arXiv preprint arXiv:2403.18120*, 2024.

Zhou, Y. From first-order logic to assertional logic. In *International Conference on Artificial General Intelligence*, pp. 87–97. Springer, 2017.

Table 4. Representative Ontology Examples

Domain	Concept	Operator
Numbers	Real	Abs, Sqrt, Log, NaturalLog, Sqrt, Cos, Sin, Tan, Get_Number_Round, Exp, Is_Real
	Integers	Get_GCD, Is_OddNumber, Get_Remainder, Get_InversedMod, Get_LCM,
	NaturalNumbers	Factorial, Get_Combination, Is_Coprime, Is_Prime, Get_Digit_Number, Get_DigitSum, Get_DigitProduct, Get_DigitCount
Polynomial	Polynomial	Get_PolyTerm, Is_PolyFactor, Get_Polyroots, Get_PolyDegree, Get_Term_Coefficient,
Function	Function	Get_Function_Range, Get_Function_Zeroes, Get_Inverse_Function, Get_Function_Value, Get_Function_Minimum, Is_Bijection
Set	Set	Set_Cardinality, Set_Union, Set_Difference, Set_Intersection, Build_Set, Get_Set_Sum, Get_Set_Maximum, Get_Set_Minimum
Sequence	Sequence	Is_GeometricSequence, Get_CommonRatio, Is_ArithmeticSequence, Get_Sequence_Sum, Get_CommonDifference
Special	———	ForAll, Exists, Get_Prod, Solve_equation, Get_Sum , Negation, Range

Appendix

A. Ontology Examples.

The complete ontology comprises only **40** concepts, **180** operators, and **6** thematic topics. In table 4, we present representative samples of core concepts and corresponding operators across topics. The full content can be accessed in our materials.

Our method incorporates the core concepts and operations from these fields. Although some concepts are not directly covered in the ontology library, they can essentially be derived through combinations of our existing operators. This demonstrates the simplicity and expressive power of the knowledge equation.

In the future, we will further expand the coverage of our ontology library to support more complex mathematical reasoning. Our plan primarily includes: **(1) Completing the ontology library to support geometric and statistical visual reasoning**, and **(2) Incorporating foundational undergraduate mathematics, such as abstract algebra, mathematical analysis, and topology**. We hope that KE can provide resources and novel insights for future research in autoformalization and theorem proving.

Table 5. Hyperparameters in all the experiments.

Type	Parameter	Value
SFT Training	Batch Size	512
	Learning Rate	$2.0e-5$
	Learning Rate Scheduler	Cosine
	Warm-up Ratio	0.01
	Optimizer	AdamW
	Epoch	15
Evaluation	Top-p	0.9
	Temperature	0.7
	Max tokens	512

Instruction:
You are an expert in the Isabelle theorem prover. Your task is to translate theorems from natural language into formal Isabelle statements. Please follow these guidelines:

1. Carefully analyze the given theorem in natural language.
2. Translate it into a correct and precise Isabelle formal statement.
3. Use the following format for your response:
```
theorem tm_name :
fixes ⟨variable⟩
assumes "⟨hypothesis⟩"
shows "⟨statement⟩"
sorry
```
4. Focus solely on the translation. Do *not* attempt to prove the theorem or provide additional explanations.
5. Ensure that your translation accurately captures all the mathematical concepts and relationships expressed in the natural language version.
6. Use appropriate Isabelle syntax, including correct use of quantifiers, implications, and mathematical symbols.
7. If the theorem involves specific mathematical structures (e.g., groups, rings, topological spaces), use the corresponding Isabelle definitions and notations.
8. Do *not* include any proofs, use `sorry` as a placeholder. Do *not* add any explanations.

The goal is to produce a syntactically correct and semantically accurate formalization in Isabelle. Your translation should faithfully reflect the meaning of the original theorem while following Isabelle conventions and best practices.

Figure 3. Instructions for Translating Natural Language into Isabelle

B. Experiments.

B.1. Training Details

We take a fully fine-tuning setting for training DeepSeek-Math-7B-Base as the base model. All training experiments are conducted on 4 NVIDIA A100 GPUs with LLaMA-Factory framework. Detailed hyperparameters utilized for training and evaluation experiments are documented in Table 5.

B.2. Prompt Details

In this section, we present all the prompts used in this work to facilitate progress. The correspondence between all prompt examples and the tables is shown below.

- Fig 3, Fig 4, Fig 5 and Fig 6, Fig 7, Fig 8 present the prompts used to translate natural language to Lean, Coq, and Isabelle respectively, with examples randomly sampled from our parallel dataset.
- Fig 9 outlines the prompt used to verify the semantic accuracy of KE during our validation phase.
- Fig 10 outlines the prompt used to guide the large language model to perform data synthesis, with templates randomly selected from our template library.
- Fig 11 demonstrates the LLM prompt used for correcting statements that fail Lean4 compilation.

Instruction:
You are an expert in the Lean4 theorem prover. Your task is to translate theorems from natural language into formal Lean4 statements. Please follow these guidelines:

1. Carefully analyze the given theorem in natural language.
2. Translate it into a correct and precise Lean4 formal statement.
3. Use the following format for your response:
   `theorem tm_name : <Lean4 formal statement> := by sorry`
4. Focus solely on the translation. Do *not* attempt to prove the theorem or provide any explanations.
5. Ensure that your translation accurately captures all the mathematical concepts and relationships expressed in the natural language version.
6. Use appropriate Lean4 syntax, including correct use of quantifiers, implications, and mathematical symbols.
7. If the theorem involves specific mathematical structures (e.g., groups, rings, topological spaces), use the corresponding Lean4 definitions and notations.
8. Do *not* include any proofs, use `sorry` as a placeholder. Do *not* add any explanations.

The goal is to produce a syntactically correct and semantically accurate formalization in Lean4. Your translation should faithfully reflect the meaning of the original theorem while following Lean4 conventions and best practices.

Figure 4. Instructions for Translating Natural Language into Lean

Instruction:
You are an expert in the Coq theorem prover. Your task is to translate theorems from natural language into formal Coq statements. Please follow these guidelines:

1. Carefully analyze the given theorem in natural language.
2. Translate it into a correct and precise Coq formal statement.
3. Use the following format for your response:
   `Theorem tm_name : <Coq formal statement>. Proof. Admitted.`
4. Focus solely on the translation. Do *not* attempt to prove the theorem or provide additional explanations.
5. Ensure that your translation accurately captures all the mathematical concepts and relationships expressed in the natural language version.
6. Use appropriate Coq syntax, including correct use of quantifiers, implications, and mathematical symbols.
7. If the theorem involves specific mathematical structures (e.g., groups, rings, topological spaces), use the corresponding Coq definitions and notations.
8. Do *not* include any proofs, use `Admitted` as a placeholder. Do *not* add any explanations.

The goal is to produce a syntactically correct and semantically accurate formalization in Coq. Your translation should faithfully reflect the meaning of the original theorem while following Coq conventions and best practices.

Figure 5. Instructions for Translating Natural Language into Coq

```
id: 1

-- Problem: Find all solutions to the equation \sqrt[3]{3 - \frac{x}{3}} = -2.

--lean_theorem:
theorem Unexplored_1 :
    { (x : ℝ ) | ( 3 - x / 3 ) ^ ( 1 / 3 ) = -2 } = sorry
    := by sorry

--coq_theorem:
Theorem Test_1 :  { x : R} | 1 / (x - 2) < 3 / x } = sorry.
    Proof.
    Admitted.

--Isabelle_theorem:
theorem Test_1 :
    shows "{ x :: real . 1 / (x - 2) < 3 / x } = sorry" sorry

id: 2

-- Problem: Given the function f(x) = |1 - 2x| - |1 + x|. Solve the inequality f(x) ≥
    4.

--lean_theorem:
theorem Unexplored_2 :
    { x : ℝ | |1 - 2 * x| - |1 + x| ≥ 4 } = sorry
    := by sorry

--coq_theorem:
Theorem Test_2 :
    { x : R | (Rabs (1 - 2 * x) - Rabs (1 + x)) >= 4 } = sorry.
    Proof.
    Admitted.

--Isabelle_theorem:
theorem Test_2 :
    shows "{x :: real. abs (1 - 2 * x) - abs (1 + x) ≥ 4} = {}"
    sorry

id: 3

-- Problem: If sequence A is an arithmetic sequence with A(1)=3, A(2)=6; find A(5)

--lean_theorem:
theorem Unexplored_3 (A : ℕ → ℝ)
    (h1 : ∃ d : ℝ, ∀ n : ℕ, A (n + 1) = A n + d)
    (h2 : A 1 = 3)
    (h3 : A 2 = 6) :
    A 5 = 15 := by sorry

--coq_theorem:
Theorem Test_3 (A : nat → R)
    (h1 : exists d, forall n, A (S n) = A n + d)
    (h2 : A 1%nat = 3)
    (h3 : A 2%nat = 6) :
    A 5%nat = 15.
Proof.
Admitted.
```

Figure 6. Few Shots for Translating Natural Language into Formal Language (Part I).

```
--Isabelle_theorem:
theorem Test_3 :
    fixes A :: "nat ⇒ real"
    assumes h1: "∃d. ∀n. A (n + 1) = A n + d"
        and h2: "A 1 = 3"
        and h3: "A 2 = 6"
    shows "A 5 = 15"
    sorry

id: 4

-- Problem: If Set M = {1, 3, 5}, Set N = {2, 3, 4}. Find the union of M and N.

--lean_theorem:
theorem Unexplored_4 (M N : Set ℝ)
    (h1 : M = {1, 3, 5})
    (h2 : N = {2, 3, 4}) :
    M ∪ N = {1, 2, 3, 4, 5} := by sorry

--coq_theorem:
Theorem Test_4 (M N : Ensemble R)
    (h1 : M = [1; 3; 5])
    (h2 : N = [2; 3; 4]) :
    Union M N = [1; 2; 3; 4; 5].
Proof.
Admitted.

--Isabelle_theorem:
theorem Test_4 :
    fixes M N :: "real set"
    assumes h1: "M = {1, 3, 5}"
        and h2: "N = {2, 3, 4}"
    shows "M ∪ N = {1, 2, 3, 4, 5}"
    sorry

id: 5

-- Problem: Solve the following equation: 5(1 - cos x) = 4 sin x

--lean_theorem:
theorem Unexplored_5 :
    {x : ℝ | 5 * (1 - Real.cos x) = 4 * Real.sin x} = sorry := by sorry

--coq_theorem:
Theorem Test_5 :
    {x : R | 5 * (1 - cos x) = 4 * sin x} = sorry.
Proof.
Admitted.

--Isabelle_theorem:
theorem Test_5 :
    shows "{x :: real. 5 * (1 - cos x) = 4 * sin x} = {}"
    sorry
```

Figure 7. Few Shots for Translating Natural Language into Formal Language (Part II).

```
id: 6

-- Problem: Given that x, y, z are positive real numbers with product xyz = 1,
-- show that the inequality holds

--lean_theorem:
theorem Unexplored_6 (x y z : ℝ)
    (hx : x > 0) (hy : y > 0) (hz : z > 0)
    (h1 : x * y * z = 1) :
    x^3 / ((1 + y) * (1 + z)) +
    y^3 / ((1 + z) * (1 + x)) +
    z^3 / ((1 + x) * (1 + y)) ≥ 3/4 := by sorry

--coq_theorem:
Theorem Test_6 (x y z : R)
    (hx : x > 0) (hy : y > 0) (hz : z > 0)
    (h1 : x * y * z = 1) :
    (x^3 / ((1 + y) * (1 + z)) +
     y^3 / ((1 + z) * (1 + x)) +
     z^3 / ((1 + x) * (1 + y))) >= 3/4.
Proof.
Admitted.

--Isabelle_theorem:
theorem Test_6 :
    fixes x y z :: real
    assumes hx: "x > 0" and hy: "y > 0" and hz: "z > 0"
        and h1: "x * y * z = 1"
    shows "x^3 / ((1 + y) * (1 + z)) +
           y^3 / ((1 + z) * (1 + x)) +
           z^3 / ((1 + x) * (1 + y)) ≥ 3/4"
    sorry
```

Figure 8. Few Shots for Translating Natural Language into Formal Language (Part III).

Instruction:
You are an expert in Lean4 language and natural language. When given a math problem described in natural language and a math problem described in Lean4 language, your task is to evaluate the consistency of the two math problems and score them.
Scoring Rules:

1. The full score is 5 points and the lowest score is 0.
2. When the semantics of all statements of the two math problems are consistent, give full marks of 5 points.
3. For each inconsistent statement, deduct 1 point until 0 points.

Response Format:

- Reply with `||your points||` in the final sentence
- Use the exact "——" format for the score

Input Format:

```
math problem described in natural language:
<ORIGINAL MATH PROBLEM>

math problem described in Lean4 language:
<LEAN4 MATH PROBLEM>
```

Output Format:

```
<SEMANTIC CONSISTENCY SCORE>
```

Figure 9. Prompt for Semantic Consistency Judgment

Instruction:
You are an expert at creating integrated math problems combining multiple concepts. When provided with knowledge **K** of operators and concepts, and labeled examples **E**, your task is to return complex math problems and their labeled results.
Rules:

1. *Never use* any new concepts or operators except those in the context!
2. Do not include any explanatory text.
3. *Strictly* follow the style of the context.
4. Combine the provided fragments effectively to create *complex* mathematical problems or proofs.
5. Return exactly 10 results in the specified format.

Output Format:

```
Problem: <problem statement>
Declaration: <required declarations>
Facts: <supporting facts>
Query: <specific question>
```

Input:

```
knowledge K:
## Concepts ##
<EXPLANATION OF CONCEPTS IN K>

## Operators ##
<EXPLANATION OF OPERATORS IN K>

labeled examples E:
<EXAMPLES OF LABELING MATH PROBLEMS>
```

Output:

```
<10 LABELED MATH PROBLEMS>
```

Figure 10. Prompt for Data Synthesis about Sequence Questions

Instruction:
You are a Lean4 expert specialized in fixing mathematical formalization errors.
Informal Statement

```
<informal>
```

Error Message

```
<error_message>
```

Incorrect Code

```
<lean_theorem>
```

Correction Rules

```
1. First identify the error type (type mismatch, syntax error, missing instance)
2. For type errors (N/Z/R/C), add explicit type annotations
3. For syntax errors, fix parentheses/commas/indentation
4. For missing instances, add required typeclass arguments
5. Output ONLY the corrected code, no explanations
```

Corrected Code

```
<Corrected Code>
```

Figure 11. Prompt for Correcting Syntactic-Error Questions

MiniF2F T29. Show that there exist real numbers a and b such that a is irrational, b is irrational, and a^b is rational.

```
theorem Unexplored_29
    (a : ℝ)(h_a : Irrational a)
    (b : ℝ)(h_b : Irrational b)
    : a ^ b ∈ ℚ
    := sorry
```

Figure 12. A formalization of MiniF2F T29 in Lean 4. $a^b \in \mathbb{Q}$ follows natural language conventions, it constitutes invalid syntax in Lean4. We note that the problem's formalization also contains inaccuracies, though our present focus remains on syntactic errors in this subsection.

C. Case Study.

This section analyzes the common types of errors in KELPS translation results, and discusses the differences between DeepSeek-V3 and Claude models in semantic scoring, focusing on typical problems at the grammatical and semantic levels. Overall, grammatical errors are relatively easy to identify, and their rules are relatively fixed, so the types of errors are limited; while the judgment of semantic consistency is more complex, often involving contextual understanding and reasoning, and is more challenging.

C.1. Syntax Errors

Grammar Errors. This might be because the model didn't see enough formal examples, or perhaps there is a discordance between formal language syntax rules and typical organic natural language patterns. A representative example is illustrated in Figure 12, where the expression $q \in \mathbb{Q}$, though common and clear in natural language, is not valid in formal syntax. The model may also adopt natural language shortcuts or formats in formal expressions (Figure 13). This could trace back to the model's exposure to abundant informal content during pre-training, while having relatively limited contact with formalized materials.

Type Errors. These errors primarily occur because, in natural language, a number/object often belongs to multiple types simultaneously, while the informal statement may be insufficient to determine its unique specific type. Generally, Lean4 supports automatic type inference, except for certain specific cases (see Figure 14). In contrast, more rigorous formal systems like Coq typically require explicit type declarations in formal statements. Version differences can also trigger such issues. Therefore, file-level (rather than theorem-level) autoformalization is essential for future research.

C.2. Semantic Errors

Misunderstanding about Informal Statement. The process of formalizing natural language math problems presents varying levels of difficulty. Since some problems lack explicit mathematical declarations and assumptions, their formalization requires first parsing the natural language semantics and then abstracting mathematical content through modeling. Unfortunately, current mainstream large language models still exhibit limitations in natural language understanding. The representative example in Figure 15 shows how models lose accuracy when faced with indirect statements.

Misalignment with Informal Statement. Other prevalent error patterns comprise omission of critical information (Fig 16) and inconsistencies with the informal statement (Fig 17). These resemble human students' typos in writing solutions, thus being nearly unavoidable. However, these relatively minor errors are theoretically preventable through self-correction and secondary checking.

FormalMath T452. Find all positive integers a, b, c such that $ab + 1$, $bc + 1$, and $ca + 1$ are all equal to factorials of some positive integers. Prove that the answer is: $\boxed{(k! - 1, 1, 1)}$ (and its permutations), where $k \in \mathbb{N}_{>1}$.

```
Declarations: "a: PositiveIntegers; b: PositiveIntegers; c: PositiveIntegers"

Facts: "a * b + 1 = Factorial(k); b * c + 1 = Factorial(k); c * a + 1 = Factorial(k)"

Query: "(a, b, c) = (k! - 1, 1, 1)",
```

Figure 13. A formalization of FormalMath T452 in Knowledge Equations. While the notation $k!$ clearly denotes the factorial function in human-readable mathematics, it fails to comply with the syntactic requirements of Knowledge Equations. Therefore, it fails to be translated into Lean4.

MiniF2F T111. Solve for a: $\frac{8^{-1}}{4^{-1}} - a^{-1} = 1$. Show that it is -2.

```
theorem Unexplored_111
    (a : ℝ)  (h1 : ( 8 : ℝ ) ^ ( -1 ) / ( 4 : ℝ ) ^ ( -1 ) - (a) ^ (-1: ℝ) = 1)
    : a = -2 := by sorry
```

Figure 14. A formalization of MiniF2F T111 in Lean4. The expression appears almost correct. However, the exponent (-1) in $(8 : \mathbb{R})^{(-1)}$ is not explicitly declared as a real number, causing Lean4 to default its type to $\mathbb{N}$ during computation and resulting in a type mismatch.

MiniF2F T198. What digit must be placed in the blank to make the four-digit integer 20_7 a multiple of 11? Show that it is 5.

```
theorem Unexplored_198
    (d : ℤ)  (h1 : 2057 % 11 = 0)
    (h2 : 2067 % 11 = 0) (h3 : 2077 % 11 = 0)
    (h4 : 2087 % 11 = 0) (h5 : d = 5)
    : 2097 % 11 = 0 := by sorry
```

Figure 15. A formalization of MiniF2F T198 in Lean4. While the generated formal statement is syntactically correct, the model fails to capture the authentic mathematical intent of the problem, consequently producing erroneous results.

MiniF2F T204. Notice that $35 \cdot 40 = 1400$ Find some integer n with $0 \leq n < 1399$ such that n is the multiplicative inverse to 160 modulo 1399. Show that it is 1058.

```
theorem Unexplored_204
    (n : ℤ)  (h1 : 0 <= n)
    (h2 : 35 * 40 = 1400) (h3 : n < 1399)
    : n = 1058 := by sorry
```

Figure 16. A formalization of MiniF2F T204 in Lean4. Although the existing content is correct, the model missed the condition that n is the multiplicative inverse to 160 modulo 1399. This renders the problem unsolvable under the given constraints.

MiniF2F T155. If a and b are real numbers, $a^2b^3 = \frac{32}{27}$, and $\frac{a}{b^3} = \frac{27}{4}$, what is $a + b$? Show that it is $\frac{8}{3}$.

```
theorem Unexplored_155
    (a : ℝ) (b : ℝ)
    (h1 : a / ( b ^ 3 ) = 27 / 4) (h2 : a * ( b ^ 3 ) = 32 / 27)
    : a + b = 8 / 3 := by sorry
```

Figure 17. A formalization of MiniF2F T155 in Lean4. The model incorrectly substituted a for a^2, resulting in misalignment with the informal statement.

C.3. Semantic Consistency Evaluation

We investigate the scoring discrepancies between DeepSeek-V3 and Claude in assessing the semantic consistency between semantic parsing outputs and the original natural language questions. Empirical results indicate that Claude consistently assigns lower scores than DeepSeek-V3, with an average difference ranging from 0.2 to 0.5 points and a maximum discrepancy of up to 1 point. An analysis of representative cases with large score gaps (see Figures 18 and 19) reveals that DeepSeek-V3 tends to be more lenient—assigning higher scores even when the outputs exhibit certain semantic deviations—whereas Claude adopts a more conservative scoring approach. Manual inspection suggests that the appropriate scores generally lie between those given by the two models, highlighting systematic differences in their scoring criteria. To mitigate potential bias introduced by individual models, we adopt the average of their scores as the final semantic consistency metric.

FormalMATH T2499. Does there exist a function $f : \mathbb{N} \to \mathbb{N}$, such that $f(f(n)) = n + 1987$ for every natural number n ?

```
theorem Unexplored_2499 :
∃ f : ℕ → ℕ, ∀ n : ℕ, f (f n) = n + 1987 := by sorry
```

Figure 19. A formalization of FormalMATH T2499 in Lean4. For the original open-ended question, the model provided a formalized hint suggesting the proposition is true after reasoning. Therefore, DeepSeek considered this formalization entirely correct, whereas Claude believed it altered the structure of the original open-ended question and thus only gave it a score of 1.

MiniF2F T50. What is the value of $\left(\sum_{k=1}^{20} \log_{5^k} 3^{k^2}\right) \cdot \left(\sum_{k=1}^{100} \log_{9^k} 25^k\right)$?
(A) 21 **(B)** $100 \log_5 3$ **(C)** $200 \log_3 5$ **(D)** 2,200 **(E)** 21,000
Show that it is **(E)** 21,000.

```
theorem Unexplored_50 :
(Σ k in Finset.range 20, Real.logb (5^k) (3^(k^2)))
* (Σ k in Finset.range 100, Real.logb (9^k) (25^k))
    = 21000 := by sorry
```

Figure 18. A formalization of MiniF2F T50 in Lean4. Because the definition of range in Lean4 starts from index 0, DeepSeek overlooked this mistake and gave a score of 5; whereas Claude considered it a serious error and only gave a score of 2.