

Automated Formalization via Conceptual Retrieval-Augmented LLMs

Wangyue Lu^{1*}, Lun Du^{2*†}, Sirui Li¹, Ke Weng¹, Haozhe Sun¹, Hengyu Liu³,
Minghe Yu¹, Tiancheng Zhang^{1†}, Ge Yu¹,

¹Northeastern University

²Ant Research Institute, Ant Group

³Department of Computer Science, Aalborg University

20223246@stu.neu.edu.cn, dulun.dl@antgroup.com,

{20226504, 2401925}@stu.neu.edu.cn, sunhaozhe@stumail.neu.edu.cn,

heli@cs.aau.dk, {yuminghe, tczhang, yuge}@mail.neu.edu.cn

Abstract

Interactive theorem provers (ITPs) require manual formalization, which is labor-intensive and demands expert knowledge. While automated formalization offers a potential solution, it faces two major challenges: *model hallucination* (e.g., undefined predicates, symbol misuse, and version incompatibility) and the *semantic gap* caused by ambiguous or missing premises in natural language descriptions. To address these issues, we propose CRAMF, a Concept-driven Retrieval-Augmented Mathematical Formalization framework. CRAMF enhances LLM-based autoformalization by retrieving formal definitions of core mathematical concepts, providing contextual grounding during code generation. However, applying retrieval-augmented generation (RAG) in this setting is non-trivial due to the lack of structured knowledge bases, the polymorphic nature of mathematical concepts, and the high precision required in formal retrieval. We introduce a framework for automatically constructing a concept-definition knowledge base from Mathlib4, the standard mathematical library for the Lean 4 theorem prover, indexing over 26,000 formal definitions and 1,000+ core mathematical concepts. To address conceptual polymorphism, we propose contextual query augmentation with domain- and application-level signals. In addition, we design a dual-channel hybrid retrieval strategy with reranking to ensure accurate and relevant definition retrieval. Experiments on miniF2F, ProofNet, and our newly proposed AdvancedMath benchmark show that CRAMF can be seamlessly integrated into LLM-based autoformalizers, yielding consistent improvements in translation accuracy—achieving up to 62.1% and an average of 29.9% relative improvement.

Introduction

Automated formalization is the process of translating natural language descriptions of mathematical theorems into formally verifiable representations (Weng et al. 2025), such as Lean (Moura and Ullrich 2021), Coq (Huet, Kahn, and Paulin-Mohring 1997), or Isabelle (Paulson 1994). In the era of large language models (LLMs), it serves as a crucial bridge between informal human reasoning and formal

symbolic logic, enabling AI systems to participate meaningfully in mathematical problem solving (Weng et al. 2025; Guo et al. 2025; Zheng et al. 2025). Its importance is exemplified by DeepMind’s AlphaProof, which achieved silver-medal-level performance in the 2024 International Mathematical Olympiad by leveraging an end-to-end formalization pipeline based on the Lean theorem prover (AlphaProof and AlphaGeometry 2024). As LLMs become central to automated theorem proving, the accuracy and reliability of automated formalization directly impact the overall success of proof generation.

Current mainstream approaches to automated formalization rely on Large Language Models (LLMs) to directly translate natural language into formal mathematical statements (Wu et al. 2022). Typical strategies include few-shot prompting of pre-trained models and fine-tuning on aligned natural language–formal language (NL–FL) pairs (Xin et al. 2024). While recent systems, such as Herald (Gao et al. 2024) and Kimina-Prover (Wang et al. 2025), have shown promising results, they continue to face two fundamental challenges. 1) **Model hallucination** arises when LLMs generate confident but incorrect formal code. Common failure modes include fabricating undefined concepts in Mathlib (Lean’s standard library), misusing symbols due to informal reasoning, and producing outdated definitions incompatible with the latest Mathlib version. 2) **Semantic gap** stems from the mismatch between the ambiguity of natural language and the precision of formal languages. A key difficulty is *conceptual polymorphism*, where identical expressions correspond to different formal definitions depending on context, domain, or abstraction level. This often leads to inaccurate formalizations, especially in applied fields like combinatorics, where essential entities are frequently implicit and hard to recover from surface text.

In general-domain natural language processing, similar challenges, such as factual hallucination and context-sensitive ambiguity, are often addressed through Retrieval-Augmented Generation (RAG) (Gao et al. 2023b). By retrieving relevant external knowledge to ground and guide model outputs, RAG has proven effective in improving factual consistency and semantic precision across a range of tasks. Despite this success, the application of RAG to

*Equal Contribution

†Corresponding Author

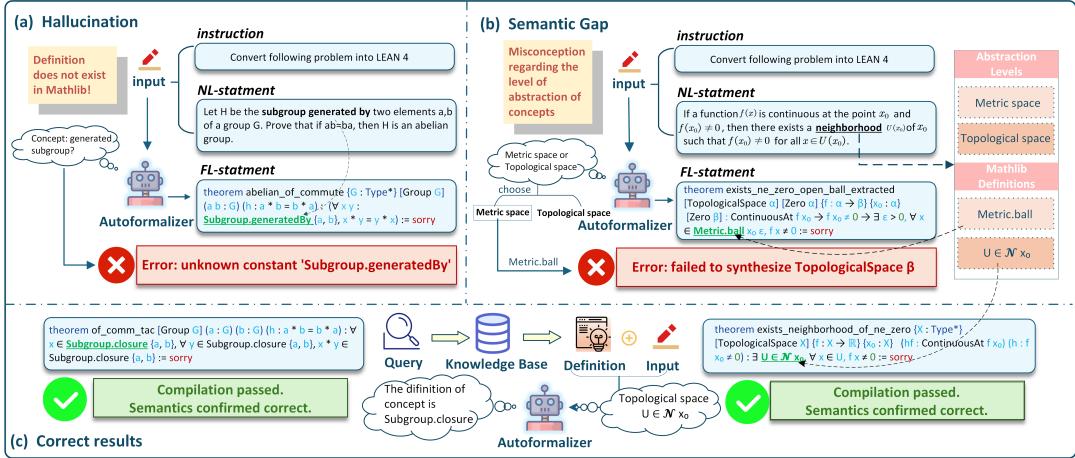


Figure 1: Examples illustrating model hallucination and semantic gap. Panel (a) demonstrates a case of autoformalization failure due to the model fabricating an undefined predicate in Mathlib, resulting in compilation errors. Panel (b) exhibits a semantic gap arising from the conceptual polymorphism of “neighborhood” (defined differently in topological versus metric spaces) where the model fails to recognize the appropriate abstraction level, triggering type class synthesis errors during compilation. Panel (c) showcases correct formalization results generated by the CRAMF framework, which effectively resolves both issues through structured knowledge grounding and context-aware concept resolution.

mathematical automated formalization remains largely unexplored. This work investigates whether retrieval-based methods can be adapted to improve the reliability and accuracy of LLM-driven formalization. However, directly applying RAG in this domain introduces new obstacles. First, unlike encyclopedic knowledge in the general domain, mathematical libraries such as Mathlib lack structured, queryable mappings from natural language expressions to formal definitions, making effective retrieval non-trivial. Second, resolving *conceptual polymorphism* requires not just retrieving related content, but disambiguating between multiple context-sensitive formalizations, a task that standard RAG pipelines are not equipped to handle. These limitations call for domain-specific retrieval augmentation tailored to the needs of formal reasoning systems.

To systematically address hallucination and semantic gap in Lean-based autoformalization, we propose the **Concept-driven Retrieval-Augmented Mathematical Formalization (CRAMF)** framework. CRAMF enhances formalization accuracy by retrieving precise definitions of core mathematical concepts from Mathlib to provide contextual grounding for LLM-based autoformalizers. At its core, CRAMF relies on a structured knowledge base that explicitly maps natural language expressions to their corresponding formal definitions. We define a schema for this concept-definition knowledge base that captures the many-to-many relationships between informal descriptions and formal representations. To support scalability and coverage, we design an automated pipeline that constructs the knowledge base by aligning Mathlib definitions with diverse, canonical natural language expressions. To address conceptual polymorphism, CRAMF augments user queries with contextual signals and leverages a hybrid retrieval strategy to improve definition disambiguation. By incorporating domain-specific cues and

application context, the system enhances its ability to distinguish between multiple candidate definitions of the same concept. A combination of symbolic and semantic retrieval, followed by reranking, ensures accurate and context-aware retrieval of formal definitions.

In conclusion, our contributions are as follows:

- We propose the **Concept-driven Retrieval-Augmented Mathematical Formalization (CRAMF)** framework, which retrieves precise formal definitions of core mathematical concepts to provide contextual grounding for LLM-based autoformalization.
- We define a structured concept-definition knowledge base covering over 26,000 Mathlib definitions and 1,000+ core mathematical concepts, and develop an automated LLM-powered pipeline to construct this resource by aligning formal definitions with diverse natural language expressions.
- We demonstrate that CRAMF serves as a plug-and-play enhancement for LLM-based autoformalizers, consistently improving translation accuracy on miniF2F (Zheng, Han, and Polu 2021), ProofNet (Azerbaiyev et al. 2023), and our proposed AdvancedMath benchmark—achieving up to 62.1% and an average of 29.9% relative improvement.

Method

This section details the proposed **Retrieval-Augmented Mathematical Formalization Framework (CRAMF)**. CRAMF follows a retrieval-augmented paradigm: given a natural language description of a mathematical theorem, it retrieves formal definitions of relevant concepts from a structured concept-definition knowledge base and provides them as contextual grounding for the LLM-based auto-

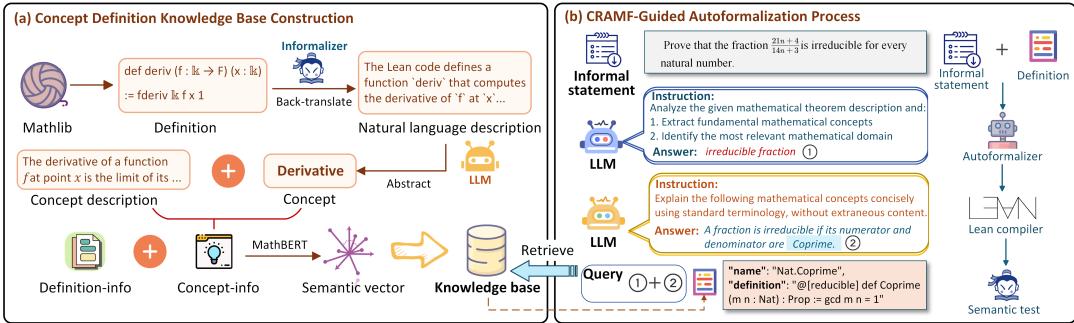


Figure 2: Overview of the CRAMF framework. (a) Construction of the concept-definition knowledge base via back-translation and concept extraction from Mathlib definitions. (b) Integration into autoformalization: extracted concepts retrieve relevant definitions to guide formal code generation.

formalizer. This enriched context helps the model generate Lean 4 code that is syntactically correct, semantically aligned with the theorem, and compliant with Mathlib specifications. We describe the framework through three main components: concept-definition knowledge base construction, mathematical concept extraction, and definition retrieval. An overview of CRAMF is illustrated in Figure 2.

Concept Definition Knowledge Base Construction

Ontology Schema Design To support structured representation and efficient retrieval of Lean-based mathematical knowledge, we define a lightweight ontology \mathcal{O} and implement a corresponding relational database schema for storage and querying. The ontology is formalized as a quadruple:

$$\mathcal{O} = (\mathcal{C}, \mathcal{P}, \mathcal{A}, \mathcal{R})$$

where \mathcal{C} is the set of entity types, \mathcal{P} is the set of attributes, $\mathcal{A} : \mathcal{C} \rightarrow \mathbb{P}(\mathcal{P})$ is the attribute mapping function that assigns each entity type its associated attributes, and \mathcal{R} is the set of semantic relations. Here, $\mathbb{P}(\mathcal{P})$ denotes the power set of \mathcal{P} .

To explicitly capture concept-definition mappings and support retrieval, we define the entity types as:

$$\mathcal{C} = \{\Gamma, \Theta, \Phi\},$$

where Γ represents abstract mathematical concepts, Θ denotes formal mathematical definitions extracted from Mathlib, and Φ consists of natural language annotations associated with those definitions. Each entity type is associated with a set of attributes via the mapping function \mathcal{A} , which serve as the logical basis for database fields: Γ is characterized by the attribute triple $\langle \gamma_n, \gamma_d, \gamma_e \rangle$, representing the concept's name, domain, and explanatory description; Θ is characterized by $\langle \theta_\tau, \theta_f, \theta_p \rangle$, representing the definition's identifier, formal expression, and module path; Φ is defined by the singular attribute ϕ_a denoting definition's natural language annotation.

The relation set \mathcal{R} defines the semantic links between entities:

$$\mathcal{R}_1 \subseteq \Gamma \times \mathbb{P}(\Phi),$$

$$\mathcal{R}_2 : \Phi \rightarrow \Theta,$$

where \mathcal{R}_1 establishes a one-to-many relationship between entity Γ and entity Θ , signifying that a single mathematical concept may correspond to multiple Lean definitions. \mathcal{R}_2 establishes a one-to-one relationship between entity Θ and entity Φ , denoting that each formal definition corresponds to exactly one unique natural language explanation.

Knowledge Base Population This subsection describes the automated population of the concept-definition knowledge base by instantiating the abstract schema with concrete records extracted from Mathlib. Guided by the attribute mapping function \mathcal{A} , we populate the three entity types $\mathcal{C} = \{\Gamma, \Theta, \Phi\}$ and establish semantic relations \mathcal{R} among them.

We begin by parsing Mathlib using Lean 4's official documentation tool, doc-gen4, to extract all definitions declared via def, class, or structure. These are used to populate the attributes of the *Definition* θ and *Description* Φ entities, including identifiers, formal representations, module paths, and associated annotations.

For instantiating the mathematical *Concept* entity Γ , we adopt a reverse translation strategy using the pre-trained language model InternLM-Math-7B (Ying et al. 2024b). Each formal definition is passed to the model to generate natural language descriptions. We apply a self-consistency validation procedure to improve generation quality, producing three candidate descriptions and selecting the one most semantically aligned with the original annotation. Subsequently, the selected natural language description is processed by a concept extraction model (DeepSeek-V3) to identify the underlying mathematical concept and generate an explanatory gloss. This final step completes the construction of the Γ entity. Illustrative examples of this end-to-end process are shown in Figure 3.

Vector Encoding and Index Construction We employ the pre-trained and fine-tuned domain-specific language model MathBERT to perform semantic encoding on the γ_e field of Γ and the θ_a field of the descriptions in the knowledge base as follows:

$$v_c = \text{MathBERT}(\gamma_e)$$

$$v_d = \text{MathBERT}(\theta_a)$$

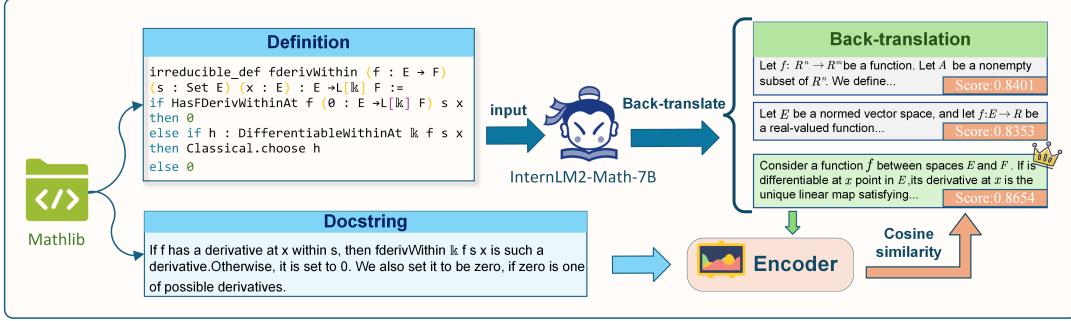


Figure 3: The example illustrates reverse translation of Lean’s defintion of *fderivWithin* via InternLM-plus7B, with semantic similarity scoring against Mathlib annotations.

An efficient vector index is then built using Faiss. Ultimately, each knowledge unit is represented as the following quadruple:

$$(\gamma_n, \theta_r, v_c, v_d)$$

where γ_n is the core mathematical concept; θ_r is its corresponding Mathlib definition name; v_c and v_d are the respective semantic vectors.

Here is the precise translation adhering to academic English conventions and the original structure:

Concept Extraction

In CRAMF, mathematical concept extraction is the core process linking natural language problems with formal definitions. The objective is to accurately identify the core mathematical concepts from the input natural language theorem description, providing fundamental query units for subsequent definition retrieval.

Within diverse mathematical expressions, some problems explicitly contain mathematical concepts, while others implicitly rely on mathematical structures. For instance, applied mathematics problems often feature concrete problem descriptions rather than abstract mathematical semantics. Therefore, we employ distinct extraction methods for different types of mathematical problems.

For conventional proof problems explicitly containing mathematical concepts: We leverage the powerful comprehension capabilities of large language models (LLMs) to directly identify and extract the core concepts within the mathematical theorem. **For applied mathematics problems with implicit mathematical concepts:** Their problem descriptions typically lack explicit mathematical concept keywords. For example, the combinatorial problem “Prove: Among any 6 people, there are always at least 3 people who either all know each other or are all strangers to each other” implicitly involves the mathematical concept of a graph. When handling such problems, we introduce an LLM-based problem rewriting mechanism. During the concept extraction phase, this mechanism performs mathematical modeling and rewriting of the original problem to explicitly express its core mathematical structure before concept extraction proceeds. Details on prompt construction and examples are provided in the appendix.

Definition Retrieval

Due to the conceptual polymorphism of mathematical concepts, the interpretation of the same naturally described concept can vary across different abstraction levels and semantic granularities. Without providing relevant context, LLMs struggle to confirm the precise formal definition corresponding to a mathematical concept when generating formal statements. Consequently, the goal of this stage is to retrieve the formal definition that best matches the mathematical concept to enhance model generation via prompting.

Query Enhancement When the same mathematical concept has different formal definitions across domains, using solely the concept as the query can lead to excessive noise in retrieval results and low recall. To address this, we employ a query enhancement method. Using LLMs under strict prompting, we perform conceptual parsing of the mathematical theorem, generating an interpretation of terms based on the extracted core concepts and the existing theorem description. The concept is then concatenated with this terminological interpretation to form the query. The model-generated interpretation incorporates domain information and application context for the concept, while also implying dependencies between concepts. Using this as the query enables matching richer semantics during retrieval, helping to mitigate inaccuracies caused by conceptual polymorphism and bridging the semantic gap between natural language descriptions and Mathlib’s specialized definitions.

Dual-Pathway Hybrid Retrieval We adopt a collaborative strategy combining hybrid retrieval and reranking, integrating semantic vector retrieval with exact matching mechanisms to ensure retrieval results satisfy both semantic relevance and formal precision. Specifically, the system executes the following two retrieval pathways in parallel:

- **Symbol-Level Keyword Matching:** The system prompts the LLM to generate search keywords for the extracted core mathematical concept. These keywords are used for exact matching via regular expressions against definition symbols within the Mathlib library, forming a base candidate set.
- **Semantic Similarity Retrieval:** The query text, composed of the concept and its interpretation, is input into

Model	MiniF2F			ProofNet			AdvancedMath		
	Base(%)	+CRAMF(%)	RG(%)	Base(%)	+CRAMF(%)	RG(%)	Base(%)	+CRAMF(%)	RG(%)
Deepseek-V3	52.3	69.3	+32.5	39.8	53.1	+33.4	31.7	48.2	+52.1
GPT-4o	70.1	82.7	+18.0	48.0	62.6	+30.4	48.5	59.1	+21.9
Herald-7B	79.1	93.6	+18.3	60.9	79.4	+30.4	80.0	90.2	+12.8
Kimina-7B	98.8	99.2	+0.4	87.4	94.1	+7.7	98.8	100	+1.2

Table 1: Compilation Pass Rate@10 of each base model (Orig.) versus the same model augmented with CRAMF on three datasets. **RG** (Relative Gain) represents the relative improvement rate.

Model	MiniF2F			ProofNet			AdvancedMath		
	Base(%)	+CRAMF(%)	RG(%)	Base(%)	+CRAMF(%)	RG(%)	Base(%)	+CRAMF(%)	RG(%)
Deepseek-V3	36.9	47.1	+27.6	23.6	37.0	+56.8	19.8	32.1	+62.1
GPT-4o	49.6	60.1	+21.2	29.9	42.2	+41.1	22.8	34.7	+52.2
Herald-7B	49.2	63.1	+28.3	44.4	55.1	+24.1	39.3	51.4	+30.8
Kimina-7B	80.3	84.8	+5.6	65.0	69.3	+6.6	61.3	66.5	+8.5

Table 2: Formalization Accuracy Rate@10 of each base model (Orig.) versus the same model augmented with CRAMF on three datasets. **RG** (Relative Gain) represents the relative improvement rate.

a MathBERT encoder to calculate its vector similarity with concept explanations in the knowledge base, initially recalling the top-10 similar concepts. These are then reranked using the bge-reranker-v2-m3 model to filter the semantically most relevant Top-5 concepts. The definitions corresponding to these concepts are merged into the base candidate set.

To further optimize prompt quality, a reranking mechanism is introduced. The bge-reranker-v2-m3 model performs fine-grained semantic assessment of candidate definitions. By calculating the similarity between the conceptual interpretation within the query and the annotations of candidate definitions, it reranks the candidates. The Top-3 definitions with the highest semantic match are ultimately selected to constitute the context prompt for the automated formalization task.

Experiments

In this section, we conduct experiments to address the following research questions:

- **RQ1:** How effectively does CRAMF framework improve compilation success rate and formalization accuracy in autoformalizing natural language mathematical theorems to Lean 4?
- **RQ2:** How does CRAMF framework perform in retrieving definitions compared to baseline methods?
- **RQ3:** What individual contributions do CRAMF’s core components make to the final formalization performance?

Experimental Setup

Datasets. We evaluate our method on two public datasets (i.e., miniF2F (Zheng, Han, and Polu 2021) and ProofNet

(Azerbayev et al. 2023)) and one proprietary dataset (AdvancedMath). AdvancedMath consists of 173 informalized proof problems in higher mathematics, constructed to benchmark autoformalization performance on advanced mathematical reasoning.

Baseline.

Autoformalization Models. We evaluate the effectiveness of the CRAMF framework against several baseline autoformalization models: the open-source autoformalizers Herald-7B (Gao et al. 2024) and Kimina-7B (Wang et al. 2025), as well as the powerful API-based LLMs DeepSeek-V3 (DeepSeek et al. 2024) and GPT-4o.

Retrieval Models. We consider three representative Retrieval-Augmented Generation (RAG) methods as baselines: BM25 (Robertson, Zaragoza et al. 2009), Rewrite-Retrieve-Read (Ma et al. 2023), and HyDE (Gao et al. 2023a).

Evaluation Methods.

Autoformalization Evaluation Method. We adopt the evaluation pipeline from the LeanWorkBook project (Ying et al. 2024a). Formalized outputs are verified by the Lean compiler. Results that compile successfully are then back-translated into natural language descriptions using the InternLM2-Math-Plus-7B model (Ying et al. 2024b). Finally, DeepSeek-V3 assesses the semantic consistency between the back-translated statements and the original informal statements.

Retrieval Performance Evaluation Method. We evaluate retrieval quality using two metrics: Average Contribution Score (ACS) and Relevant Definition Hit Rate (HitRate@K).

1) Average Contribution Score (ACS) measures the overall relevance of retrieved definitions to the target problem. Each

Method	MiniF2F	ProofNet	AdvancedMath
BM25	0.91	0.94	0.83
R3	1.38	1.45	1.31
HyDE	1.55	1.61	1.52
CRAMF	2.07	2.14	1.93

Table 3: Average Contribution Scores (ACS) of retrieval methods on three datasets. Scores range from 0 to 3; higher is better.

Method	MiniF2F	ProofNet	AdvancedMath
BM25	11.3%	14.5%	9.1%
R3	19.6%	21.1%	15.4%
HyDE	32.4%	35.5%	30.7%
CRAMF	44.2%	50.6%	42.9%

Table 4: Top-3 hit rate of relevant definitions (HitRate@3) for each retrieval framework across three datasets.

retrieved definition is assigned a score from 0 to 3 based on its contribution to the formalization process. A score of **3 (Exact Match)** indicates that the definition appears in the final formalized code, compilation succeeds, semantic consistency assessment passes, and the generated Lean 4 expression closely aligns with the original problem. A score of **2 (Strong Relevance)** is given when the definition is not used in the final code but is semantically or mathematically related to the problem statement, as judged by a large language model. A score of **1 (Weak Relevance)** is assigned when the definition shares topical relevance but lacks direct semantic connection (also evaluated by an LLM). A score of **0 (Erroneous Reference)** indicates that the definition appears in the code but either fails to compile or fails the semantic consistency assessment.

Scores of 3 and 0 are automatically determined using regular expression matching on the formalized output. Scores of 2 and 1 are assessed using DeepSeek-R1; see the Appendix for prompts. Given a sample set $T = \{t_1, t_2, \dots, t_n\}$, where each problem t_i has a set of retrieved definitions $\mathcal{D}_i = \{d_{i1}, d_{i2}, \dots, d_{ik}\}$, the ACS is defined as:

$$ACS = \frac{1}{\sum_{i=1}^n |\mathcal{D}_i|} \sum_{i=1}^n \sum_{j=1}^{|\mathcal{D}_i|} \text{Score}(d_{ij}, t_i) \quad (1)$$

where $\text{Score}(d_{ij}, t_i) \in \{0, 1, 2, 3\}$ denotes the contribution level of definition d_{ij} to problem t_i .

2 Relevant Definition Hit Rate (HitRate@K) measures whether at least one high-quality definition appears among the top- K retrieved results for each problem. Specifically, we check whether any of the top- K definitions achieve a score of 2 or higher. Formally, this metric is defined as:

$$HitRate@K = \frac{1}{n} \sum_{i=1}^n \mathcal{I} \left[\max_{j=1, \dots, K} \text{Score}(d_{ij}, t_i) \geq 2 \right] \quad (2)$$

where n is the total number of problems and $\mathcal{I}[\cdot]$ is the indicator function (1 if the condition holds, 0 otherwise). In

Method	MiniF2F	ProofNet	AdvancedMath
CRAMF	63.1%	55.1%	51.4%
w/o MCP	58.2%	49.8%	44.0%
w/o DCR	48.5%	36.8%	37.1%
w/o Rerank	54.3%	47.5%	43.9%

Table 5: Ablation results of CRAMF submodules evaluated by FAR@10.

Method	CombMath
CRAMF	63.1%
w/o ReWrite	54.3%

Table 6: Effect of structured problem rewriting on FAR@10 over 241 combinatorics problems in CombMath.

our experiments, we set $K = 3$. This metric captures the proportion of problems for which the top- K retrieved definitions contain at least one that is strongly relevant or an exact match.

Autoformalization Performance (RQ1)

We evaluate the effectiveness of the CRAMF framework in enhancing various automatic formalization models using two metrics: Compilation Pass Rate@10 (CPR@10) and Formalization Accuracy Rate@10 (FAR@10). The comparative results are presented in Table 1 and Table 2. CRAMF consistently improves the performance of all baseline models, although the extent of improvement varies depending on the underlying model architecture.

Particularly notable are the substantial gains observed in the general large model DeepSeek-V3 without domain-specific fine-tuning, indicating that CRAMF effectively fills formalization knowledge gaps in general-purpose models through precise definition retrieval, constructing a “plug-and-play” formal knowledge bridge that substantially lowers the domain entry barrier. On the AdvancedMath dataset involving multiple complex concepts, the relative improvement rates for both compilation pass rate and formalization accuracy reach their highest values at 52.1% and 62.1% respectively. The significant improvement in compilation pass rate confirms that injecting core mathematical concept definitions enables large models to express formal symbols with greater precision, while the enhancement in formalization accuracy reflects deeper semantic comprehension of mathematical problems.

Knowledge Base and Retrieval Performance (RQ2)

We first evaluate the performance of the CRAMF in accurately retrieving core mathematical definitions, aiming to verify the effectiveness of its constructed concept-definition knowledge base and hybrid retrieval strategy in the task of automatic formalization of mathematical theorems. Table 3 and Table 4 report the average contribution score (ACS) and the top-3 hit rate (HitRate@3) of different retrieval frameworks across three datasets. As shown in both tables, CRAMF significantly outperforms all baselines, achieving

an average improvement of more than 0.5 points in contribution score and up to a 15-percentage-point increase in HitRate@3 over the best-performing baselines. Notably, on AdvancedMath dataset—characterized by a high proportion of complexly phrased concepts—CRAMF maintains robust performance, demonstrating strong generalization ability under complex mathematical contexts.

In contrast, BM25 relies on surface-level keyword matching, making it difficult to distinguish between the semantic variations of the same concept across different levels of abstraction. As a result, it tends to introduce considerable noise. The Rewrite-Retrieve-Read method performs better than BM25, indicating that rewriting improves the recall of implicit concepts. However, it still fails to meet the symbolic precision requirements of Lean. Although HyDE shows improved ACS compared to the first two baselines, it suffers from hallucination issues and may lead to retrieving incorrect definitions, resulting in suboptimal retrieval performance compared to our method.

Ablation Study (RQ3)

To analyze the contributions of individual submodules within the CRAMF framework, particularly considering the condition-triggered problem rewriting mechanism in our retrieval pipeline, we conduct ablation experiments in two parts. 1) On three general datasets (i.e., miniF2F, ProofNet, and AdvancedMath), we evaluate the impact of three core components: Mathematical Concept Parsing (MCP), Dual-Channel Retrieval (DCR), and reranking. 2) We assess the effectiveness of the rewriting mechanism in addressing problems with high representational abstraction and implicit terminology using CombMath, a custom-built dataset of 241 combinatorics problems derived primarily from the textbook “*Combinatorics: The Art of Counting*” (Sagan 2020), an authoritative source covering core subdomains such as enumerative combinatorics, combinatorial design, and algebraic combinatorics. We use Herald-7B (Gao et al. 2024) as the backbone generation model. Table 5 reports the ablation results on the general datasets. Removing the MCP module results in a 6–7 percentage point drop in FAR, showing that providing domain knowledge and application contexts of mathematical concepts plays a crucial role in disambiguating definitions and improving retrieval precision. Removing the dual-channel retriever causes the most substantial degradation in performance, with average FAR dropping by 15.7%, respectively. This indicates that in symbol-rich and syntactically diverse environments like Lean, a single retrieval strategy is insufficient to balance semantic relevance and symbolic precision. The re-ranking module also proves essential—due to high noise in the initially recalled definitions, removing this component leads to a performance drop of 7–9%. Table 6 presents the results of analyzing the rewriting mechanism on CombMath. Enabling the rewriting module improves FAR by 8.8%, validating the practicality and necessity of our conditionally triggered rewriting strategy for complex semantic modeling tasks.

Related Works

Autoformalization

Autoformalization refers to the process of transforming informal mathematical statements into formally verifiable representations (Jiang, Li, and Jamnik 2023; Poiroux et al. 2024; Wang et al. 2020). Existing approaches can be broadly categorized into rule-based and LLM-based methods. Rule-based methods leverage explicit logical and syntactic rules (Weng et al. 2025), often adopting controlled natural languages (e.g., Mizar (Rudnicki 1992), ForTheL (Vershinich and Paskevich 2000)) or grammatical frameworks (e.g., GF (Pathak 2024)) to construct abstract syntax trees that are then translated into formal code. In contrast, LLM-based approaches utilize few-shot prompting (Wu et al. 2022) or fine-tuning on aligned natural language–formal language (NL–FL) pairs (Xuejun et al. 2025). Despite recent progress, autoformalization remains hindered by key challenges, including model hallucination, semantic ambiguity, logical inconsistency (Li et al. 2024), data scarcity (Wu et al. 2024), and inherent limitations in current model capabilities. Retrieval-Augmented Generation as emerged as a promising solution to these issues by enabling dynamic access to relevant external knowledge for contextual grounding.

Retrieval-Augmented Generation

Retrieval-Augmented Generation (RAG) enhances LLM outputs by integrating external knowledge retrieved in response to the input query, thereby reducing hallucinations and factual errors (Li, Yuan, and Zhang 2024; Zhang et al. 2024; Liao, Chen, and Du 2023; He et al. 2024). In the context of formal reasoning, RAG has been applied to mathematical autoformalization and verification tasks. Azerbayev et al. (2023) propose a retrieval-augmented formalization method that selects prompt-relevant statements to improve proposition generation. RAutorormalizer (Liu et al. 2025) adopts retrieval-based augmentation to support statement-level formalization. Similarly, ReProver (Yang et al. 2023) incorporates a premise selection module that filters relevant auxiliary statements from large formal corpora to aid theorem generation. While prior RAG-based autoformalization have considered retrieving relevant statements or premises, they often lack structured modeling of mathematical concepts and fail to address contextual disambiguation challenges. In contrast, our work introduces a concept-level retrieval framework grounded in a structured knowledge base, enabling more precise alignment between natural language descriptions and formal definitions.

Conclusion

This paper proposes CRAMF, a Concept-driven Retrieval-Augmented Mathematical Formalization framework. By automatically constructing a Mathlib4 concept definition knowledge base and incorporating query augmentation, multi-channel hybrid retrieval, and re-ranking mechanisms, CRAMF effectively suppresses model hallucinations and semantic gaps. Experimental results demonstrate that CRAMF significantly improves Lean4 compilation success rates and

formalization accuracy across multiple benchmarks and advanced mathematical tasks, validating its robustness and generalization capability in high-precision mathematical formalization scenarios. Future work may extend to more complex mathematical domains and explore cross-library, multi-step reasoning, and feedback-incorporated retrieval augmentation strategies.

References

- AlphaProof, T.; and AlphaGeometry, T. 2024. AI achieves silver-medal standard solving International 178 Mathematical Olympiad problems. *DeepMind blog*, 179: 45.
- Azerbayev, Z.; Piotrowski, B.; Schoelkopf, H.; Ayers, E. W.; Radev, D.; and Avigad, J. 2023. ProofNet: Autoformalizing and formally proving undergraduate-level mathematics (2023). *arXiv preprint arXiv:2302.12433*.
- DeepSeek, A.; Liu, A.; Feng, B.; Xue, B.; Wang, B.; Wu, B.; Lu, C.; Zhao, C.; Deng, C.; Zhang, C.; et al. 2024. Deepseek-v3 technical report. *arXiv preprint arXiv:2412.19437*.
- Gao, G.; Wang, Y.; Jiang, J.; Gao, Q.; Qin, Z.; Xu, T.; and Dong, B. 2024. Herald: A natural language annotated lean 4 dataset. *arXiv preprint arXiv:2410.10878*.
- Gao, L.; Ma, X.; Lin, J.; and Callan, J. 2023a. Precise zero-shot dense retrieval without relevance labels. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 1762–1777.
- Gao, Y.; Xiong, Y.; Gao, X.; Jia, K.; Pan, J.; Bi, Y.; Dai, Y.; Sun, J.; Wang, H.; and Wang, H. 2023b. Retrieval-augmented generation for large language models: A survey. *arXiv preprint arXiv:2312.10997*, 2(1).
- Guo, J.; Yang, W.; Zhang, S.; Xu, T.; Du, L.; Zheng, D.; and Huang, Z. 2025. Right Is Not Enough: The Pitfalls of Outcome Supervision in Training LLMs for Math Reasoning. *arXiv preprint arXiv:2506.06877*.
- He, X.; Zhou, M.; Xu, X.; Ma, X.; Ding, R.; Du, L.; Gao, Y.; Jia, R.; Chen, X.; Han, S.; et al. 2024. Text2analysis: A benchmark of table question answering with advanced data analysis and unclear queries. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, 18206–18215.
- Huet, G.; Kahn, G.; and Paulin-Mohring, C. 1997. The coq proof assistant a tutorial. *Rapport Technique*, 178: 113.
- Jiang, A. Q.; Li, W.; and Jamnik, M. 2023. Multilingual mathematical autoformalization. *arXiv preprint arXiv:2311.03755*.
- Li, J.; Yuan, Y.; and Zhang, Z. 2024. Enhancing llm factual accuracy with rag to counter hallucinations: A case study on domain-specific queries in private knowledge-bases. *arXiv preprint arXiv:2403.10446*.
- Li, Z.; Wu, Y.; Li, Z.; Wei, X.; Zhang, X.; Yang, F.; and Ma, X. 2024. Autoformalize mathematical statements by symbolic equivalence and semantic consistency. *Advances in Neural Information Processing Systems*, 37: 53598–53625.
- Liao, J.; Chen, X.; and Du, L. 2023. Concept understanding in large language models: An empirical study.
- Liu, Q.; Zheng, X.; Lu, X.; Cao, Q.; and Yan, J. 2025. Rethinking and improving autoformalization: towards a faithful metric and a dependency retrieval-based approach. In *The Thirteenth International Conference on Learning Representations*.
- Ma, X.; Gong, Y.; He, P.; Zhao, H.; and Duan, N. 2023. Query rewriting in retrieval-augmented large language models. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, 5303–5315.
- Moura, L. d.; and Ullrich, S. 2021. The Lean 4 theorem prover and programming language. In *Automated Deduction—CADE 28: 28th International Conference on Automated Deduction, Virtual Event, July 12–15, 2021, Proceedings 28*, 625–635. Springer.
- Pathak, S. 2024. GFLean: An autoformalisation framework for lean via GF. *arXiv preprint arXiv:2404.01234*.
- Paulson, L. C. 1994. *Isabelle: A generic theorem prover*. Springer.
- Poiroux, A.; Weiss, G.; Kunčak, V.; and Bosselut, A. 2024. Improving autoformalization using type checking. *arXiv preprint arXiv:2406.07222*.
- Robertson, S.; Zaragoza, H.; et al. 2009. The probabilistic relevance framework: BM25 and beyond. *Foundations and Trends® in Information Retrieval*, 3(4): 333–389.
- Rudnicki, P. 1992. An overview of the Mizar project. In *Proceedings of the 1992 Workshop on Types for Proofs and Programs*, 311–330.
- Sagan, B. E. 2020. *Combinatorics: The art of counting*, volume 210. American Mathematical Soc.
- Vershinin, K.; and Paskevich, A. 2000. ForTheL—the language of formal theories. *International Journal of Information Theories and Applications*, 7(3): 120–126.
- Wang, H.; Unsal, M.; Lin, X.; Baksys, M.; Liu, J.; Santos, M. D.; Sung, F.; Vinyes, M.; Ying, Z.; Zhu, Z.; et al. 2025. Kimina-prover preview: Towards large formal reasoning models with reinforcement learning. *arXiv preprint arXiv:2504.11354*.
- Wang, Q.; Brown, C.; Kaliszyk, C.; and Urban, J. 2020. Exploration of neural machine translation in autoformalization of mathematics in Mizar. In *Proceedings of the 9th ACM SIGPLAN International Conference on Certified Programs and Proofs*, 85–98.
- Weng, K.; Du, L.; Li, S.; Lu, W.; Sun, H.; Liu, H.; and Zhang, T. 2025. Autoformalization in the Era of Large Language Models: A Survey. *arXiv preprint arXiv:2505.23486*.
- Wu, Y.; Jiang, A. Q.; Li, W.; Rabe, M.; Staats, C.; Jamnik, M.; and Szegedy, C. 2022. Autoformalization with large language models. *Advances in neural information processing systems*, 35: 32353–32368.
- Wu, Z.; Wang, J.; Lin, D.; and Chen, K. 2024. Lean-github: Compiling github lean repositories for a versatile lean prover. *arXiv preprint arXiv:2407.17227*.
- Xin, H.; Guo, D.; Shao, Z.; Ren, Z.; Zhu, Q.; Liu, B.; Ruan, C.; Li, W.; and Liang, X. 2024. Advancing theorem proving in LLMs through large-scale synthetic data. In

The 4th Workshop on Mathematical Reasoning and AI at NeurIPS'24.

Xuejun, Y.; Zhong, J.; Feng, Z.; Zhai, P.; Yousefzadeh, R.; Ng, W. C.; Liu, H.; Shou, Z.; Xiong, J.; Zhou, Y.; et al. 2025. Mathesis: Towards Formal Theorem Proving from Natural Languages. *arXiv preprint arXiv:2506.07047*.

Yang, K.; Swope, A.; Gu, A.; Chalamala, R.; Song, P.; Yu, S.; Godil, S.; Prenger, R. J.; and Anandkumar, A. 2023. Leandojo: Theorem proving with retrieval-augmented language models. *Advances in Neural Information Processing Systems*, 36: 21573–21612.

Ying, H.; Wu, Z.; Geng, Y.; Wang, J.; Lin, D.; and Chen, K. 2024a. Lean workbook: A large-scale lean problem set formalized from natural language math problems. *Advances in Neural Information Processing Systems*, 37: 105848–105863.

Ying, H.; Zhang, S.; Li, L.; Zhou, Z.; Shao, Y.; Fei, Z.; Ma, Y.; Hong, J.; Liu, K.; Wang, Z.; et al. 2024b. Internlm-math: Open math large language models toward verifiable reasoning. *arXiv preprint arXiv:2402.06332*.

Zhang, Y.; Sharma, K.; Du, L.; and Liu, Y. 2024. Toward mitigating misinformation and social media manipulation in llm era. In *Companion Proceedings of the ACM Web Conference 2024*, 1302–1305.

Zheng, D.; Du, L.; Su, J.; Tian, Y.; Zhu, Y.; Zhang, J.; Wei, L.; Zhang, N.; and Chen, H. 2025. Knowledge augmented complex problem solving with large language models: A survey. *arXiv preprint arXiv:2505.03418*.

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