Multi-Level Knowledge Graph for Formal Mathematics

O. M. Ataeva,[1, *] A. P. Khalov,[1, 2, **] and N. P. Tuchkova[1, *]**

(Submitted by S. A. Lurie)

[1] *Federal Research Center "Computer Science and Control", Russian Academy of Sciences, Moscow, 119333 Russia*
[2] *Moscow Institute of Physics and Technology (National Research University), Dolgoprudny, Moscow Region, 141701 Russia*

Received April 12, 2026; revised April 17, 2026; accepted April 22, 2026

Abstract—We present a multi-level knowledge graph for formal mathematics built on the SciLib^{RU} ontology. This ontology is modular and is formalized in OWL and description logic. It strictly distinguishes three levels of description: the interpretation level, which captures what an object means; the representation level, which describes how this object is expressed; and the resource level, which specifies where the object is stored and in what form it exists. This separation yields an addressee-invariant knowledge space suitable for both human researchers and AI agents. The knowledge graph is materialized from the filtered Mathlib library which is a formalized mathematical library implemented in the Lean 4 proof assistant. It consists from approximately 190 000 statements with typed dependency edges and a domain taxonomy of 660 subclasses with provenance metadata. A multimodal embedding model trained on five types of mathematical objects correctly matches elements from different modalities 74% of the time on the first try. We demonstrate the practical utility of the approach through combined vector-graph structured lemmas retrieval for automated theorem proving with LLM DeepSeek-proover-v2 7B. On the MiniF2F benchmark, which includes 488 tasks and 50,752 runs, our system significantly outperforms three state-of-the-art search engines for Lean: LeanSearch, LeanFinder, and LeanExplore. All observed improvements are statistically significant. Our vector search and LeanSearch produce statistically indistinguishable results, with a p-value close to 1. This confirms that the observed improvement is driven by the use of graph structure, specifically the symbolic layer, as well as ontology-based hint retrieval and categorization, rather than by embedding quality alone.

2010 Mathematical Subject Classification: `68T30, 68T50, 03B35, 68P15`

Keywords and phrases: *knowledge graph, ontology, description logic, formal mathematics, Lean 4, Mathlib, multimodal representation, premise retrieval, automated theorem proving*

1. INTRODUCTION

Formalization and structuring of scientific knowledge remains a central challenge in computer science [1]. The development of formal proof languages (Lean 4 [2], Coq, Isabelle), computational algebra systems, and generative models creates a demand for *semantic libraries*. They presents knowledge representation systems that connect semantics, syntactic forms, and physical data in a manner invariant to the addressee which can be a human researcher or an AI agent. This work continues the line of research on formalization of scientific knowledge in digital libraries LibMeta [3, 4].

Existing approaches can be divided into three directions. (1) *Ontologies of scientific communication* as SPAR [5], ORKG [6]. They describe publication metadata but do not model the semantics of statements. (2) *Ontologies of mathematical knowledge* OntoMath^{PRO} [7], OMDoc [8], MMT/Math-in-the-Middle [9]. They fix terminology and taxonomy but do not link concepts to formal code or physical data. (3) *Search systems for formal libraries* as LeanSearch [10], LeanExplore [11],

* E-mail: oataeva@frccsc.ru
** E-mail: khalov.a@phystech.edu
*** E-mail: ntuchkova@frccsc.ru

LeanFinder [12]. They provide retrieval and reproducibility but introduce no symbolic semantic layer. Analysis of these directions reveals three open problems:

1. No separation between the *meaning* of a mathematical object, its *form*, and its *data*. Mathematical object can be presented in textual, formulaic, code-based *form* which contains *data* in files, embeddings, storage coordinates.

2. No support for *multiple coequal representations* of a single object: the same theorem can be expressed in natural language, in LaTeX notation, in Lean 4 code, or as a rendered formula image, and all forms must be treated equally.

3. No model of *contextual truth*: existing ontologies treat truth as an absolute property, whereas the same statement may be true in one axiomatic system and false in another.

Our approach is based on a hypothesis of knowledge connectivity and an representation invariant design principle. *Hypothesis of knowledge connectivity:* mathematical knowledge is inherently connected across the boundaries of individual documents; the dependency structure of a knowledge graph (which lemmas a theorem uses in its type and proof, which math domain their shares and how they can be traced to core lemmas or axioms) captures this connectivity and can serve as a navigational structure for both human and machine agents. *Design principle (representation invariance):* the meaning of a mathematical object can and should be separated from any particular representation; the resulting invariant structure (shared URI linking interpretation, representations, and resources) reduces noise and simplifies the construction of a neuro-symbolic search space.

In this work, we address the three open problems above and test out statements above experimentally. Our contributions are:

1. **The SciLibru$^{\mathbf{RU}}$ ontology** – a modular OWL/DL ontology with a strict three-level semantic separation into interpretation, representation, and resource. The ontology formalizes contextual truth and introduces AI-annotation provenance.

2. **A knowledge graph of the mathematical domain** materialized from Mathlib: ∼190 000 statements, typed dependency edges, 660 domain subclasses, 66M RDF triplets, vector store extensions with near one million data objects vectorized from 5 different modalities.

3. **Experimental validation:** graph-structured premise retrieval (named mode C21) statistically significantly outperforms three state-of-the-art Lean search engines on 488 MiniF2F tasks wuth total 50 752 runs across 19 experimental setups.

The paper is organized as follows. Section 2 reviews related work and formalize hypothesis and design principles. Section 3 describes the SciLib ontology. Section 4 presents knowledge graph construction. Section 5 introduces the multimodal embedding model. Section 6 contains experimental evaluation. Sections 7–9 discuss results, limitations, and conclusions.

2. RELATED WORK

2.1. Formalization of scientific knowledge at scale

Formalization of scientific knowledge is developing along two parallel tracks. The *document-centric* track treats knowledge as collections of texts with metadata: Semantic Scholar [13] indexes 205M+ publications with SPECTER embeddings and citation graphs; ORKG [6] decomposes papers into *problem–method–result* triples; SPAR [5] standardizes publication metadata. These systems operate at impressive scale but do not model the internal semantics of scientific statements.

The *concept-centric* track models knowledge as connected mathematical objects rather than documents. OntoMath$^{\mathrm{PRO}}$ [7] provides a taxonomy of more than 3400 mathematical concepts linked to DBpedia and MSC classification. OMDoc [8] partially separates content such as mathematical structures from presentation or formatting. MMT/Math-in-the-Middle [9] introduces an intermediate level between formal systems (Coq, Lean, Isabelle). The semantic library

LibMeta [14] formalizes the ontology of a digital library's content, defining it through types of resources, their attributes, and information objects; Ataeva, Serebryakov, and Tuchkova [3, 4] developed the semantic space "Mathematics" within this framework.

Our work continues the LibMeta line, extending it with a strict three-level separation (interpretation / representation / resource) and binding to formal proof libraries. The three-level structure is conceptually related to the FRBR model (Work / Expression / Manifestation) adopted in library science, specialized here for scientific objects with formal code.

2.2. LLMs and formal theorem proving

Large language models have achieved remarkable results in mathematical reasoning. Two paradigms coexist: *informal* (text generation checked by experts or benchmarks) and *formal* (automatic verification via proof assistants).

The formal paradigm relies on languages such as Lean 4 [2], whose library Mathlib implicitly contains interconnected mathematical knowledge without binding to specific scientific texts. The existence of an external deterministic verifier that checks proofs in finite time has enabled GRPO-style reinforcement learning: DeepSeekMath [15] introduced the GRPO mechanism; DeepSeek-Prover-V1.5 [16] added proof assistant feedback and Monte-Carlo tree search; DeepSeek-Prover-V2 [17] combined subgoal decomposition with RL, reaching 88.9% on miniF2F (671B model).

Alongside specialized models, complex proving infrastructures have emerged: Kimina-Prover [18] interleaves informal reasoning with Lean code via RL at 72B scale; Aristotle [19] achieved gold-medal performance at IMO 2025 through Monte-Carlo graph search and lemma decomposition; HILBERT [20] combines recursive reasoning with FAISS-based retrieval over Mathlib (99.2% on miniF2F). These systems explore vast proof-search spaces (sampling budgets of 8000+ attempts, hours of wall time). In user-facing scenarios assisting a working mathematician as fast, deterministic, traceable retrieval is preferable to exhaustive sampling.

2.3. Premise retrieval for Lean 4

Premise retrieval provides relevant lemmas to a prover model as contextual hints. LeanSearch [10] informalizes Mathlib theorems via GPT-3.5 and retrieves by embedding similarity. LeanFinder [12] fine-tunes embeddings on user-intent queries with RLHF. LeanExplore [11] combines embeddings, BM25+ lexical matching, and PageRank from the dependency graph. Herald [21] provides a natural-language-annotated Lean 4 corpus used by multiple retrieval systems. LeanDojo-v2 [22] offers an end-to-end framework for data extraction, fine-tuning, and proof search.

All these systems return *flat lists* of lemma signatures without indicating *how* to apply each lemma (which tactic: `apply`, `rw`, and `simp`). A symbolic semantic layer – ontological structure, typed relations, tactic annotations – is absent.

2.4. Multimodal contrastive learning

Contrastive learning aligns representations from different modalities in a shared space. The paradigm progresses from pairwise losses (contrastive loss [23], InfoNCE [24], SimCLR [25]) through dual-encoder architectures (CLIP [26]: text + image, 400M pairs) to multi-view extensions: Contrastive Multiview Coding [27] shows that representation quality improves with the number of views and provides the "full graph" formulation $\mathcal{L}_F = \sum_{i<j} \mathcal{L}(V_i, V_j)$ for M views with $\binom{M}{2}$ pairwise losses.

Mathematical objects can be (and often do) expressed (represented) in $M > 2$ modalities (e.g., natural language, Lean 4 code, LaTeX formula, and image). However, pairwise alignment for $M > 3$ scales as $O(M^2)$; centroid-based alignment – aggregating modalities into a single invariant representation – scales as $O(M)$. Existing contrastive frameworks do not address the problem of defining the abstract object that manifests across modalities: they align representations but do not separate meaning from form.

2.5. Positioning and problem statement

We do not build a search engine for Lean 4 and do not propose a new theorem-proving method. We propose a *neuro-symbolic semantic library* in which a typed knowledge graph provides the symbolic semantic layer, and multimodal embeddings provide the neural layer. Both layers are connected through URIs and combined during structured retrieval.

Existing systems either provide a taxonomy (OntoMath$^{\text{PRO}}$), or search (LeanSearch, LeanFinder, LeanExplore), or vector representations (CLIP-style embeddings), or complex sampling infrastructure (Kimina, Aristotle, HILBERT), but none provides a multi-level semantic graph that simultaneously connects formal objects, domains, typed dependencies, and tactic-annotated hints suitable for fast, deterministic AI-assisted scenarios. In this paper we show, within the domain of mathematics, why a dataset and an embedder alone are insufficient for meaningful work with mathematical knowledge.

Positioning: We do not build a search engine for Lean 4 and do not propose a new method for automated theorem proving. We propose a *neuro-symbolic knowledge space* in which a typed knowledge graph provides the symbolic semantic layer, and multimodal embeddings provide the neural layer. Both layers are connected through URIs and combined during structured retrieval. The mathematical domain serves as a demonstration of the approach.

Existing systems either provide a taxonomy (OntoMath$^{\text{PRO}}$), or search (LeanSearch, LeanFinder), or vector representations (embedding models), but none provides a multi-level semantic graph that simultaneously connects formal objects, domains, semantic properties, and annotations suitable for AI scenarios. In this paper we show, within a single domain, why a dataset and an embedder alone are insufficient for meaningful work with mathematical knowledge.

Statement: *W*e can formalize hypothesis of this paper and main ontological design principle as follows: *Hypothesis (knowledge connectivity).* Define the typed dependency neighborhood of an entity $e \in \texttt{Entity}^{\mathcal{I}}$:

$$N(e) = \{e' \in \texttt{Entity}^{\mathcal{I}} \mid \texttt{usesInType}^{\mathcal{I}}(e, e') \ \vee \ \texttt{usesInValue}^{\mathcal{I}}(e, e')\},$$

and its k-step expansion $N^{(\leq k)}(e) = \bigcup_{j=0}^{k} N^{(j)}(e)$. The hypothesis states that for a query entity e_q with formal statement $\varphi(e_q)$ and a generative model $\mathcal{G}$,

$$P\big(\mathcal{G} \vdash \varphi(e_q) \bigm| \varphi(e_q),\ \mathrm{repr}(N^{(\leq k)}(e_q))\big) \ > \ P\big(\mathcal{G} \vdash \varphi(e_q) \bigm| \varphi(e_q)\big)$$

for small k and tasks where the bare model has low success rate. Here $\mathrm{repr}(S)$ denotes the set of representations of entities in S, and $\mathcal{G} \vdash \varphi$ means that the model generates a proof verified by Lean. This hypothesis is tested directly in the experiment (Section 6, C21 vs A0, $p < 0.001$ on hard tasks).

Design principle (representation invariance). For an entity e, let $R(e) = \{r \in \texttt{Representation}^{\mathcal{I}} \mid \texttt{hasRepresentation}^{\mathcal{I}}(e, r)\}$. The principle requires: (a) all interpretive properties (`studiedIn`, `isSpecializationOf`, `hasTruthAssessment`) are defined on e, not on any $r \in R(e)$; (b) no representation is canonical: $\neg\,\exists\, f\colon \texttt{Entity}^{\mathcal{I}} \to \texttt{Representation}^{\mathcal{I}}$ such that $f(e)$ is privileged over $R(e) \setminus \{f(e)\}$. As a consequence, queries at the interpretation level (e.g., SPARQL over `usesInType`) return results independently of the query modality.

3. THE SCILIB ONTOLOGY: SEMANTIC MODULE

Following the classical definition of ontology as a formal specification of a shared conceptualization [28], the SciLib ontology is formalized in a subset of OWL 2 DL. The constructs actually used are restricted to the fragment $\mathcal{ALCHIQ}(\mathcal{D})$: atomic negation ($\mathcal{AL}$), full concept conjunction and disjunction ($\mathcal{C}$), role hierarchy ($\mathcal{H}$), inverse roles ($\mathcal{I}$), qualified cardinality restrictions ($\mathcal{Q}$), and concrete domains ($\mathcal{D}$) for datatype properties.

Let $\Sigma = (\mathbf{C}, \mathbf{R}, \mathbf{I})$ be a DL signature, where $\mathbf{C}$ is a set of concept names, $\mathbf{R}$ is a set of role names, $\mathbf{I}$ is a set of individual names. An interpretation $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$ assigns a nonempty domain $\Delta^{\mathcal{I}}$ and a function $\cdot^{\mathcal{I}}$ mapping each concept $C \in \mathbf{C}$ to $C^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}}$, each role $R \in \mathbf{R}$ to $R^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$, and each individual $a \in \mathbf{I}$ to $a^{\mathcal{I}} \in \Delta^{\mathcal{I}}$. The DL theory of SciLib is $\mathcal{T}_{\text{SciLib}} = (\mathcal{T}_{\text{Box}}, \mathcal{A}_{\text{Box}})$, where $\mathcal{T}_{\text{Box}}$ is

the terminological part (axioms about concepts and roles) and $\mathcal{A}_{\text{Box}}$ is the assertional part (facts about concrete objects).

The ontology[4] is modular, comprising three groups of modules: *infrastructure* (Core, Bridge, Full), *process* (Social, System, Scholarly), and *semantic* (Interpretation, Representation, Resource). In this paper we focus on the semantic group, which constitutes the main contribution. All cross-module axioms reside in the Bridge module, ensuring independent evolution of semantic modules (Fig. 1).

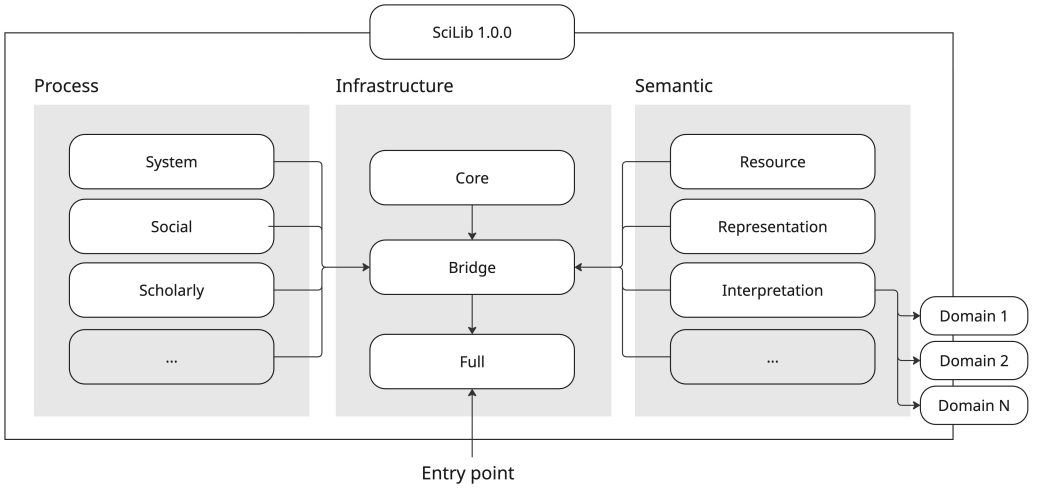

Figure 1. Modular architecture of the SciLib ontology. The semantic group (right) contains three layers of abstraction. All cross-module axioms reside in Bridge. Full is the entry point importing all modules. Arbitrary domain can be attached to Interpretation layer.

3.1. Interpretation

The Interpretation module formalizes the acts of interpreting scientific knowledge and fixing the context in which an object of knowledge is assigned meaning, epistemic status, and a truth relation.

3.1.1. Domains and entities. The class `Domain` represents a subject area. For mathematics, the hierarchy contains 660 subclasses extracted from the Mathlib directory nomenclature and annotated by GPT-5.2. Each annotation carries provenance metadata: `aiAuthor` (model name), `aiReasoning` (chain of reasoning), `aiAnalysisDate` (date), `confidence` $\in [0, 1]$. This design enables filtering and progressive replacement of AI-generated annotations with expert-validated ones.

Domain subsumption is expressed via `rdfs:subClassOf`:

$$\texttt{GroupTheory} \sqsubseteq \texttt{AbstractAlgebra} \sqsubseteq \texttt{Math}.$$

The class `Entity` describes content objects. Subclasses of `Statement` include `Axiom`, `Theorem`, `Lemma`, `Hypothesis`, `Conjecture`. The class `Proof` is linked to `Statement` by the role `hasProof`. For axioms, the ontology imposes a restriction:

$$\forall x \, \big(\texttt{Axiom}(x) \to \neg \, \exists p \ \texttt{hasProof}(x, p)\big),$$

an axiom is accepted within a context without proof; the existence of a proof turns the statement into a theorem.

The link between an entity and a domain is established via the role `studiedIn` (inverse: `hasEntityStudiedIn`). This is an interpretive link, not `rdf:type`: it may be multiple and context-dependent.

[4] SciLib ontology public release. https://github.com/andkhalov/scilib$_o ntology$

3.1.2. Semantic properties. Three semantic properties capture relations between domain *instances* (not between classes, to avoid confounding taxonomic and interpretive semantics):

- `isSpecializationOf` – transitive, irreflexive, asymmetric;
- `isGeneralizationOf` – inverse of the above;
- `isIsomorphicTo` – symmetric, transitive, irreflexive.

These properties are defined between instances of `Domain`, not between classes themselves, to preserve the possibility of contextual interpretation and to avoid committing class-level subsumption to specific semantic relationships.

3.1.3. Contextual truth. Following Tarski's semantic conception [29], truth in SciLib is not an absolute property of a statement but a relation between a statement and a context: the parallel postulate is true in Euclidean geometry and false in the geometry of Lobachevskii. The ontology models truth as a partial function $tv\colon \mathcal{E} \times \mathcal{C} \rightharpoonup \mathcal{V}$ (entity × context ⇀ truth value), reified via the class `TruthAssessment` that links entity, context, and truth value as a ternary relation (OWL/DL supports only binary relations). While the present experiment does not exercise truth assessments directly, this mechanism is a structural requirement under the open-world assumption (OWA): any knowledge graph integrating statements from multiple formal systems must distinguish the context in which a statement holds.

3.2. Representation

The Representation module describes forms of knowledge expression independently of semantic content and storage. Conceptually (an informal characterization, not a DL definition), a representation is a pair:

$$\texttt{Representation} \longleftrightarrow \langle \textit{modality},\ \textit{spec} \rangle,$$

where *modality* is the kind of expression and *spec* is the format specification. In OWL this is expressed as existential restrictions: $\texttt{Representation} \sqsubseteq \exists \texttt{hasModality.Modality} \sqcap \exists \texttt{hasSpec.Spec}$.

Modalities form a hierarchy: `TextModality` (subclasses: `NaturalLanguage`, `FormalLanguage`), `VisualModality`, `AudioModality`. For formal languages, the classes `ExecutionContext` (`toolVersion`, `platform`, `containerImage`) and `BuildRecipe` (`selector`, `template`) ensure reproducibility: a text in a formal language (Lean, Python, Java) is not self-contained, as its correct interpretation depends on the version of tools, platform, imports, and dependencies.

Key principle: all representations of a single object are *coequal*. There is no canonical form – there is a set of equivalent representations linked by a shared URI of the root entity in the Interpretation layer. This prevents conflating meaning with form, an assumption common in systems where *Meaning* ≡ *Text* or *Knowledge* ≡ *Embedding*.

3.3. Resource

The Resource module describes material and computable carriers of knowledge – data, embeddings, machine learning models, and storage systems – independently of semantic content and form. Conceptually (not a DL definition), a data resource is characterized by four attributes

$$\texttt{Data} \longleftrightarrow \langle T,\ V,\ H,\ L(s) \rangle,$$

where T is the data type, V is the version, H is the checksum, and $L(s)$ is the set of locators (storage coordinates in backend s). In OWL: $\texttt{Data} \sqsubseteq \exists \texttt{hasDataType.DataType} \sqcap \exists \texttt{storedIn.StorageBackend}$.

`StorageBackend` (PostgreSQL, S3/MinIO, Qdrant, GraphDB) is separated from `DataLocation` (specific coordinates: bucket, collection, named graph etc.). This permits migrating data between backends without changing the semantics.

Embeddings are not a modality. An embedding is a derived computable resource, not a form of expression. The class `ModalityEmbedding` (linked to `Model` via `producedBy`) represents an embedding of a specific modality. The class `DerivedEmbedding` (attribute `derivationMethod`) describes aggregated embeddings, including the centroid. Treating embeddings as resources rather than representations prevents the confusion between proximity in vector space and semantic equivalence.

3.4. *Why three levels*

The three-level separation (interpretation / representation / resource) is the central design principle of SciLib and its key distinction from existing ontologies. It enables three capabilities simultaneously:

1. *Representation invariance.* A single mathematical object (e.g., Entity "Cauchy–Schwarz inequality" see Fig. 2) exists as an abstract entity at the interpretation level with multiple coequal representations (Informal natural language form, Lean 4 code, LaTeX formula, rendered image). Queries at the interpretation level cross representation boundaries, while queries at the representation level remain format-specific.

2. *Neuro-symbolic integration.* The symbolic layer (graph edges, typed relations) and the neural layer (embeddings, similarity search) are unified through the ontology: typed edges belong to the interpretation level, embeddings belong to the resource level, and both reference the same entity via URI.

3. *Simplified graph structure.* All meta-structure is attached to the abstract entity at the interpretation level; representations and resources inherit it by reference, avoiding redundant cross-links.

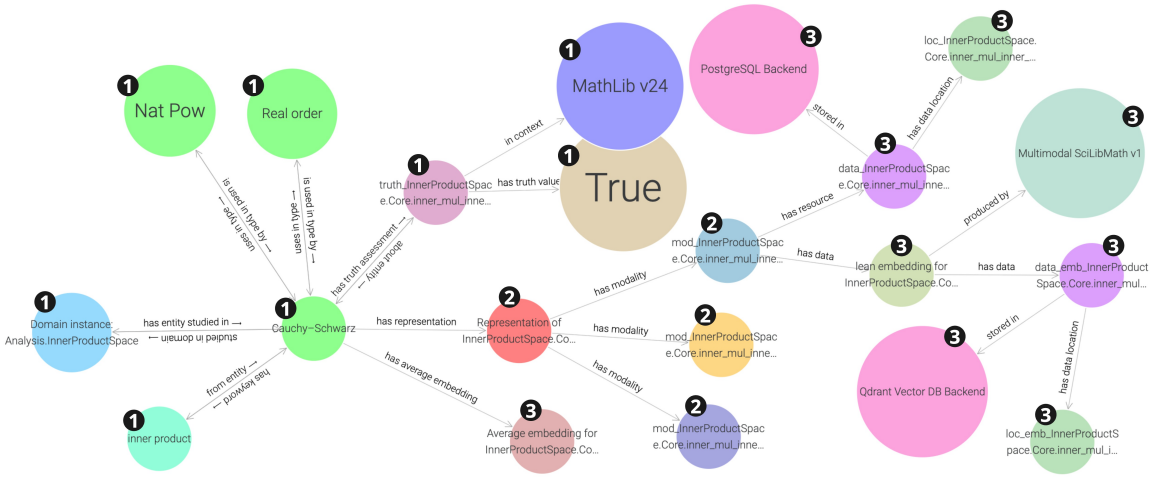

Figure 2. Entity (Cauchy–Schwarz inequality): a mathematical object described in knowledge graph by SciLib ontology (only two modalities shown). Different conceptual layers marked as: (1) Interpretation, (2) Representation, (3) Resource.

4. KNOWLEDGE GRAPH CONSTRUCTION

We describe the construction of the knowledge graph as a sequence of stages, indicating at each stage where semantic properties and AI annotations appear.

Stage 1. Data source. Mathlib [2] is the largest library of formalized mathematics for Lean 4. The parser JIXIA processes compiled Mathlib files and generates declaration files (`.decl.json`) and symbol files (`.sym.json`) containing types, proof bodies, and dependencies for each statement. [5]

Stage 2. Metadata extraction. For each statement we extract: name, module, kind (theorem, lemma, definition, axiom), attributes (`@[simp]`, `@[ext]`, etc.), Lean 4 code, and dependencies. The

[5] JIXIA: A static analysis tool for Lean 4. https://github.com/frenzymath/jixia

result is stored in PostgreSQL (table `mathlib_statements`, 213 338 records). At this stage we have *raw data from Mathlib* without any ontological interpretation.

Stage 3. Ontology application. Each statement is typed as an instance of the corresponding SciLib ontology class (`Theorem`, `Lemma`, `Definition`, etc.). Two types of edges are extracted from dependencies: `usesInType` (use in type signature) and `usesInValue` (use in proof body). These typed edges are a fundamental distinction from simple dependency graphs: they distinguish "A mentions B in its signature" (A is likely a consequence of B) from "A uses B in its proof" (B is a technical tool for A).

Stage 4. Domain taxonomy and semantic properties. The domain taxonomy is extracted from the Mathlib directory structure. Each directory (e.g., `Mathlib.GroupTheory`) becomes a subclass of `Domain`. For each of the 660 subclasses, GPT-5.2 generates: a description, the parent domain, and semantic properties (`isSpecializationOf`, `isGeneralizationOf`). All annotations carry provenance: `aiAuthor = "gpt-5.2"`, `confidence` $\in [0, 1]$, `aiReasoning` (chain of reasoning). This layer is *AI-generated* and requires expert validation; the provenance metadata enables progressive refinement.

Stage 5. RDF materialization. All data are serialized as RDF triples and loaded into GraphDB (RDF triplestore). The resulting graph contains approximately 190 000 entity nodes, 660 domain class nodes, and typed edges.

Resulting knowledge graph. This methodology transformed trasform MathLib implicit structure into 33.8M explicit RDF triplets and 35.9M inferred triplets by Hermit reasoner in the GraphDB. Final graph contain over 66M triplets.

Example: the Cauchy–Schwarz inequality. The theorem `cauchy_schwarz_aux` (module `Mathlib.Analysis.InnerProductSpace.Defs`) is represented at all three levels of the ontology. At the *interpretation* level: the entity has type `Statement`, is linked to its domain via `studiedIn` → `Analysis.InnerProductSpace`, and carries typed dependency edges: `usesInType` → `InnerProductSpace.Core.normSq`, `InnerProductSpace.Core.toPreInner'`; `usesInValue` → `mul_assoc`, `mul_left_comm`, `InnerProductSpace.Core.inner_conj_symm`, `inner_smul_left`, and others. At the *representation* level: `hasRepresentation` links the entity to its Lean 4 code, LaTeX formula, and natural-language descriptions. At the *resource* level: `hasAverageEmbedding` links to the centroid embedding in Qdrant. All three levels reference the same URI of the entity in interpretation layer.

Stage 6. Integration with the embedding model. The multimodal embedding model (Section 5) operates at the resource level: embeddings are stored in Qdrant and linked to entity URIs through the `ModalityEmbedding` class. The centroid embedding (Section 5, Eq. 1) is stored as a `DerivedEmbedding`. During retrieval, graph-based candidates (symbolic layer, interpretation level) are augmented with vector-based candidates (neural layer, resource level), both referencing the same entities via URIs.

5. MULTIMODAL EMBEDDING MODEL

The knowledge graph is complemented by a multimodal embedding model that represents each of 5 modalities of a mathematical object (English text, Russian text, Lean 4 code, LaTeX notation, formula image) as a vector in a shared 312-dimensional space. For each object o_i with modality embeddings $\{e_i^1, \ldots, e_i^M\}$, $M = 5$, the centroid

$$c_i = \frac{1}{M} \sum_{m=1}^{M} e_i^m \tag{1}$$

serves as a modality-invariant representation of the object.

The base text encoder is SciRus Tiny 3.5 [30] цшер 312-dimensional output, the visual encoder uses overlapping patches processed by ConvNeXt and projected into the same space. The dataset SciLibModal v2 consists of ∼972 000 objects from Mathlib and was used for contrastive training on base of pairwise InfoNCE [24] and centroid-based alignment. On full training (100% data, 15 epochs), the centroid-based leave-one-out retrieval on ∼19 000 held-out objects achieves R@1 > 0.99 which denotes the proportion of correct relevant objects retrieved at rank 1 for all text

modalities and R@1 = 0.93 for images (the weakest direction). Cross-modal search in the same 19 K data fold shown cm@1 = 0.74 denotes the proportion of correct cross-modal correspondences retrieved at rank 1. We use cosine similarity as a closure metric.

In the terms of the SciLib ontology, embeddings are objects of the Resource layer: `ModalityEmbedding` (linked to `Model` via `producedBy`), with the centroid stored as a `DerivedEmbedding`. The URI links each embedding to `Entity` and `Domain` through the Bridge module.

6. EXPERIMENTAL EVALUATION

To demonstrate the practical utility of the ontology-driven approach, we evaluate graph-structured premise retrieval for automated theorem proving.

6.1. Setup

Benchmark: MiniF2F [31], 488 tasks (Test: 244, Valid: 244), formalized in Lean 4. Using both splits is justified: neither the model nor the knowledge graph were trained on MiniF2F tasks.

Model: DeepSeek-Prover-V2-7B [17] (bfloat16, not fine-tuned).

Parameters: $T = 0.6$, top_p $= 0.95$, $K = 8$ attempts per (task, mode) pair.

Verifier: Lean 4 REPL, timeout 30 s, maxHeartbeats 2M.

Total volume: 50 752 runs (488×7 modes $\times\, 8 +$ ablation).

Metric: pass@k is defined as

$$\text{pass@}k = \frac{1}{N}\sum_{i=1}^{N}\left[1 - \frac{\binom{n_i - c_i}{k}}{\binom{n_i}{k}}\right],$$

where N is the number of tasks, n_i is the number of samples for task i, c_i is the number of correct samples [31]. At $k = 1$ this reduces to the empirical frequency of success.

Statistical test. We use the Wilcoxon signed-rank test [32] (paired, two-sided). For each task i, the per-task pass rate $r_i^{(X)} = c_i^{(X)}/n_i$ is computed for modes X and Y, and the paired differences $d_i = r_i^{(X)} - r_i^{(Y)}$ are formed. The test statistic is

$$W = \sum_{i:\, d_i \neq 0} \operatorname{sign}(d_i)\, R_i,$$

where R_i is the rank of $|d_i|$ among all nonzero $|d_j|$. The null hypothesis H_0: the median of d_i is zero (the two modes are equally effective). The p-value is the probability of observing a test statistic at least as extreme as W under H_0.

We prefer the Wilcoxon test over parametric alternatives (e.g., paired t-test) for two reasons: (a) pass rates are discrete (taking values $0, \frac{1}{8}, \frac{2}{8}, \ldots, 1$) and their distribution is strongly bimodal, violating the normality assumption; (b) the test operates on ranks rather than absolute values, making it robust to outliers. Significance levels: * $p < 0.05$, ** $p < 0.01$, *** $p < 0.001$.

Stratification. Tasks are divided into strata by the bare-model score (A0). *Hard:* A0 pass@1 $\leq 25\%$ (232 tasks). *Partial-capability zone:* A0 solves 1–4 out of 8 (59 tasks). *Easy:* A0 $> 50\%$ (241 tasks).

6.2. The C21 pipeline

Our primary mode, C21, combines graph-based and vector-based retrieval with zero LLM calls for seed generation. The pipeline consists of 10 steps: (1) regex-based feature extraction (types: `Nat`, `Int`, `Real`; operations: `dvd`, `pow`, `le`, etc.); (2) pattern classification against 9 predefined mathematical patterns; (3) seed resolution via SPARQL in GraphDB; (4) graph expansion along typed edges (`usesInType`: up to 20 neighbors; `usesInValue`: up to 15); (5) PostgreSQL enrichment (Lean code, attributes, module); (6) candidate classification by tactic usage (`apply`, `rw`, `simp`, `def`); (7) domain-specific simp lemma search; (8) vector augmentation via Qdrant; (9) structured formatting with tactic headers; (10) model generation.

The critical distinction from all baselines is Step 6: each hint is annotated with the tactic the model should use. The model receives not only *what* to use but also *how*:

```
-- Useful theorems (use with apply):
-- dvd_trans
@[trans] theorem dvd_trans : a | b -> b | c -> a | c

-- Simp lemmas (use with simp [...]): dvd_refl, ...
```

Baselines provide flat lists of signatures without tactic context.

Role of multimodal embeddings. In Step 8, the vector search in Qdrant uses *centroid* embeddings (Eq. 1) — aggregated representations of the mathematical object across all five modalities. Although the query is a Lean 4 statement, the search operates in a modality-invariant space: the centroid captures information from all representations (English, Russian, LaTeX, image), not just from the Lean modality. This is the multimodal component of the neuro-symbolic pipeline.

6.3. Baselines

Three state-of-the-art Lean search engines:

LeanSearch [10] performs informalization with GPT-3.5 → embedding → cosine similarity.

LeanFinder [12] relies on user-intent fine-tuned embeddings combined with RLHF alignment..

LeanExplore [11] employs a hybrid ranking pipeline that integrates embedding based retrieval, BM25 search and PageRank signals derived from the dependency graph.

All baselines receive 10 hints in an identical prompt template (flat list of `name : type`).

We additionally consider three internal configurations: *A0*, which uses the bare model without any external hints; *B1*, which applies SciLib's vector-based retrieval implemented with Qdrant; and *C23*, which combines all graph-based retrieval strategies with a subsequent LLM-driven re-ranking stage.

6.4. Results

Table 1 reports results for the following retrieval and baseline modes: *C21*, which combines graph-based and vector-based retrieval and serves as our primary configuration; *C23*, which augments graph-based retrieval with a re-ranking stage; *B1*, which relies exclusively on vector search; A0, which uses the bare model without any retrieval; *BL_LS*, corresponding to the LeanSearch engine; *BL_LF*, corresponding to LeanFinder; and *BL_LE*, corresponding to LeanExplore

Table 1. pass@1 by difficulty stratum (488 tasks)

Mode	All (488)	Hard (232)	Partial (59)	Easy (236)
C21	**50.0%**	**8.6%**	**48.9%**	89.4%
C23	48.9%	8.0%	40.9%	88.5%
BL_LF	48.8%	7.3%	37.7%	88.9%
A0	48.7%	3.3%	28.8%	**93.5%**
B1	48.2%	7.4%	41.1%	87.4%
BL_LS	47.9%	6.2%	37.9%	88.1%
BL_LE	47.0%	5.7%	30.1%	87.2%

Across all tasks, the differences between modes remain relatively small, with pass@1 ranging from 47.0% to 50.0%. This compression is expected because ∼50% are consistently solved (easy) and ∼39% are never solved with A0 achieving 0 successful attempts out of 8. The partial-capability zone contains 59 tasks and is the most informative: C21 nearly doubles pass@1 compared to A0 increasing pass@1 from 28.8% to 48.9% and outperforms the strongest baseline LeanSearch by +11 percentage points with 48.9% compared to 37.9%. The statistical significance of these differences is further supported by the Wilcoxon signed-rank tests reported in Table 2. As shown in Figure 3 C21 leads across all slices, with the largest advantage observed in the partial-capability zone. This effect is examined in more detail in Figure 4, which focuses specifically on the partial-capability zone.

Table 2. Wilcoxon tests (p-value): C21 vs all modes

C21 vs	All (488)	Hard (232)	Partial (59)
LeanSearch (BL_LS)	*0.032	0.079	*0.031
LeanFinder (BL_LF)	0.100	0.386	*0.012
LeanExplore (BL_LE)	**0.001	0.027*	***0.001
SciLib vector (B1)	*0.024	0.391	0.059
Bare model (A0)	0.163	< ***0.001	***0.001

* $p < 0.05$, ** $p < 0.01$, *** $p < 0.001$.

6.5. Analysis

The graph is the differentiator, not embedding quality. B1 (SciLib vector search) and BL_LS (LeanSearch) yield statistically identical results ($p = 0.931$). Both are pure vector search systems with different embedding models. Since C21 significantly outperforms B1 ($p = 0.024$), the entire C21 advantage is explained by the graph structure and tactic-aware hint categorization, not by superior embeddings.

Typed edges outperform PageRank. LeanExplore uses PageRank from the dependency graph — a graph-like signal. Yet LeanExplore is the weakest baseline (47.0%). PageRank yields a single scalar "importance" score but does not structure information for the model. Typed edges (`usesInType`/`usesInValue`) and tactic classification provide qualitatively richer context.

Pattern-based seeds outperform model-generated seeds. In the ablation study (exp. 140, 244 tasks, 16 modes), C11 (regex patterns, 0 LLM calls, ∼8 s) outperforms C1 (model-generated seeds, ∼30 LLM calls, ∼163 s) by +2.8 pp on hard tasks (8.8% vs 6.0%). Deterministic regex rules are more precise and 11× faster than neural seed generation for navigating the dependency graph.

AIME: a category where baselines lead. On the 30 AIME tasks, LeanFinder (5.4%) and LeanSearch (5.0%) outperform C21 (3.8%). AIME problems often require non-standard tricks

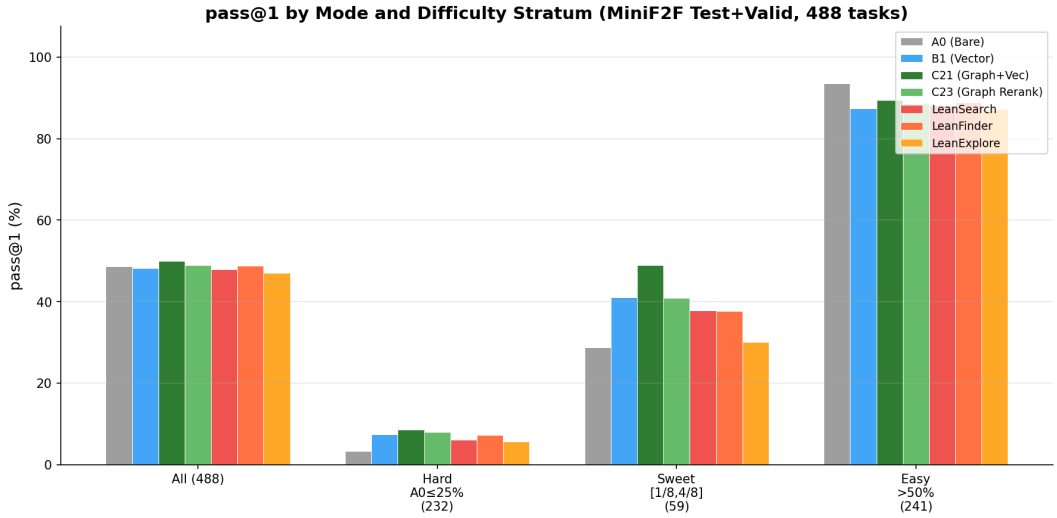

Figure 3. pass@1 by A0 difficulty stratum. C21 leads across all slices; the advantage is maximal in the partial-capability zone.

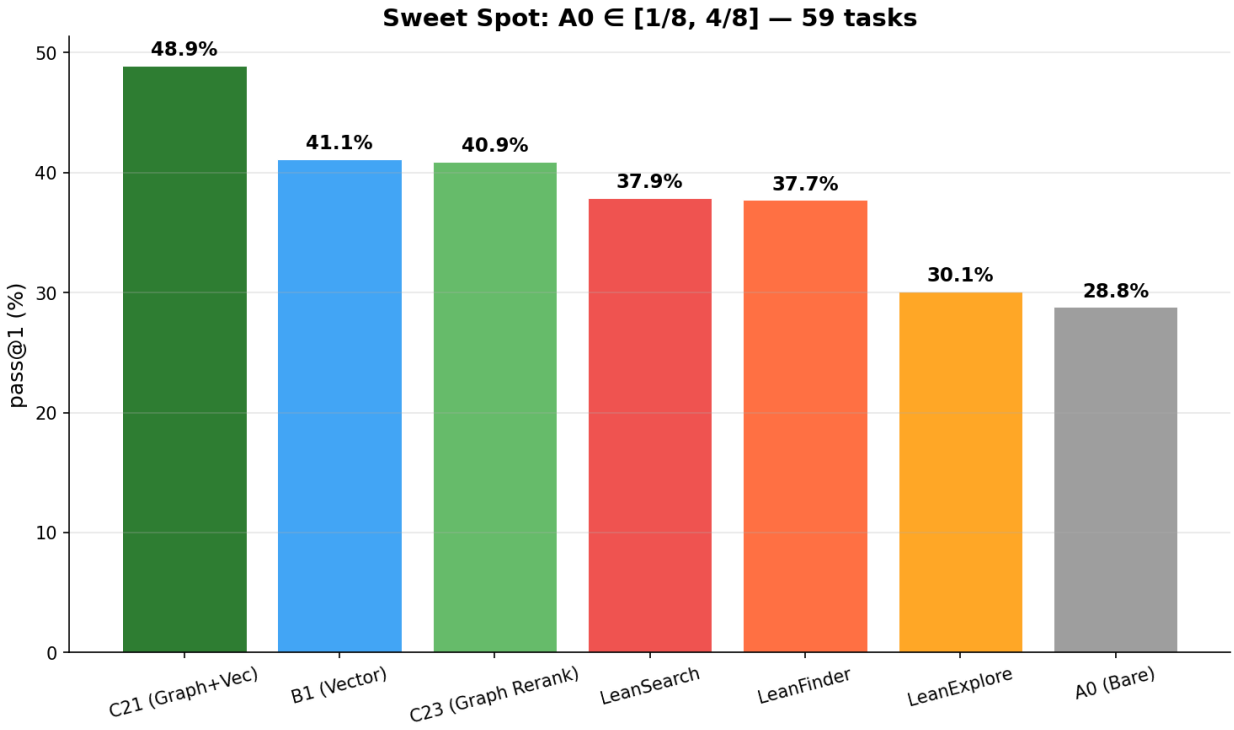

Figure 4. Partial-capability zone analysis. On tasks where the bare model solves 1–4 out of 8 attempts, C21 achieves 48.9% pass@1 vs 28.8% for A0 and 37.9% for LeanSearch.

that are better captured by informalized semantic search than by dependency-graph traversal. This result indicates that C21 and vector search are complementary rather than substitutive.

Exclusive solutions. On hard tasks in the Test split, graph-based modes solve 4 tasks that no baseline solves, while baselines solve only 1 task that no graph mode solves (ratio 4:1). The graph expands the set of solvable tasks, not merely increases the probability of solving already-solvable ones.

6.6. Case study

In the task `mathd_numbertheory_320`, the objective is to prove $n = 34$ from the hypotheses $n < 101$ and $101 \mid (123456 - n)$. A0 solves 1/8, attempting `rfl`, which fails; LeanSearch solves 2/8, returning relevant but tactically unannotated lemmas. By contrast, C21 solves 8/8: the regex extractor identifies `Nat.dvd`, graph expansion retrieves `dvd_refl` and `dvd_trans`, tactic classification assigns them the labels `simp` and `apply`, and the model generates `norm_num at h1; interval_cases n <;> omega` (PASS, 434 ms). These structured hints direct the model toward arithmetic tactics rather than a direct proof of equality.

7. DISCUSSION

The SciLib ontology addresses the three gaps identified in the introduction: (1) three-level separation of interpretation, representation, and resource; (2) multiple coequal representations; (3) contextual truth formalization.

The experimental results demonstrate that the graph structure and ontology-driven hint categorization – not embedding quality – are the source of the C21 advantage. The equivalence of B1 and LeanSearch ($p = 0.931$) provides direct evidence: any vector search, regardless of the embedding model, reaches approximately the same ceiling on this benchmark. The graph provides what embeddings cannot: typed dependencies and tactic annotations.

It is important to trace the causal chain from the ontology to the experimental result. Without the three-level model, there would be no typed edges (`usesInType`/`usesInValue`) — only an undifferentiated dependency graph. Without typed edges, there would be no tactic-aware classification of candidates (Step 6 of the C21 pipeline), because the distinction between "used in type signature" and "used in proof" is precisely what enables classifying a lemma as `apply` vs `rw` vs `simp`. Without tactic classification, C21 would reduce to flat-list retrieval – statistically indistinguishable from LeanSearch ($p = 0.931$). The entire experimental advantage thus originates in the ontological design decision.

The knowledge graph is not a theorem solver. It serves as a structured vocabulary, providing names of tactics and lemmas relevant to the problem formulation. The solution remains with the generative model. The ontology ensures traceability: each hint traces back to a specific graph edge (`usesInType` or `usesInValue`).

The neuro-symbolic architecture – symbolic graph traversal at the interpretation level combined with neural similarity search at the resource level – is unified through the ontology. This design is addressee-invariant: both a human mathematician and an AI prover can query the same knowledge graph for contextually grounded information.

8. LIMITATIONS

A discussion of the limitations is necessary for the correct interpretation of the presented results.

One important limitation is that the study is confined to a single subject domain (mathematics), which makes validation in other scientific areas necessary. Owing to resource constraints, the taxonomy of 670 domain classes was generated by an LLM and requires expert validation. In addition, with only $K = 8$ attempts, the analysis retains limited statistical power at the individual-task level. Also the study uses a single generative model (DeepSeek-Prover-V2-7B), which further limits the scope of the evaluation. The use of 9 pattern categories implies incomplete coverage of task types. Moreover the system is tied to Lean4/Mathlib, and extension to other formal systems was not tested. In the multimodal model, images remain the weakest modality with R@1 ≈ 0.61 vs > 0.82 for text modalities. Furthermore, the results of LeanSearch and LeanFinder depend on the availability of external APIs, and we report results obtained during February–March 2026.

9. CONCLUSION

We present the SciLib ontology as a modular OWL/DL ontology that implements a three-level separation into interpretation, representation and resource layers and provides a formalization of contextual truth.

A knowledge graph for the mathematical domain is constructed from Mathlib, comprising approximately 213 000 statements connected by typed dependency edges and 670 domain subclasses enriched with AI-generated annotations and provenance metadata. A multimodal embedding model covering five modalities achieves a top-1 cross-modal matching accuracy of 0.74 when using centroid-based aggregation.

On the MiniF2F benchmark, which includes 488 tasks and 50 752 runs, graph-structured premise retrieval (C21) shows a statistically significant advantage over three state-of-the-art Lean search engines. The observed equivalence of vector search methods, with a p-value of 0.931, indicates that the performance gains are attributable to the graph structure and tactic-aware categorization rather than to differences in embedding quality.

Future work will focus on extending the approach to additional scientific domains, validating the AI-generated taxonomy with expert review, expanding the set of pattern categories, and integrating the system with interactive proof workflows.

10. FUNDING

This work was supported by budget topics of the Ministry of Science and Higher Education of the Russian Federation 'Mathematical methods for data analysis and forecasting'.

11. CONFLICT OF INTEREST

The authors declare that they have no conflicts of interest.

REFERENCES

[1] K. Yang, G. Poesia, J. He, W. Li, K. Lauter, S. Chaudhuri, and D. Song, "Formal mathematical reasoning: A new frontier in AI", Proc. 42nd Int. Conf. on Machine Learning (ICML 2025), PMLR **267**, 82384–82398 (2025). https://proceedings.mlr.press/v267/yang25az.html

[2] L. de Moura and S. Ullrich, "The Lean 4 theorem prover and programming language", Automated Deduction – CADE 28: 28th Int. Conf. on Automated Deduction, LNCS **12699**, 625–635 (Springer, 2021). https://doi.org/10.1007/978-3-030-79876-5_37

[3] O. M. Ataeva, V. A. Serebryakov, and N. P. Tuchkova, "Development of the semantic space 'Mathematics' by integrating a subspace of its applied area", Lobachevskii J. of Mathematics **43** (12), 3435–3446 (2023). https://doi.org/10.1134/S1995080222150069

[4] V. A. Serebryakov and O. M. Ataeva, "Ontology-based approach to modeling of the subject domain 'Mathematics' in the digital library", Lobachevskii J. of Mathematics **42**, 1920–1934 (2021). https://doi.org/10.1134/S199508022108028X

[5] S. Peroni and D. Shotton, "The SPAR ontologies", The Semantic Web – ISWC 2018, Lecture Notes in Computer Science **11137**, 119–136 (2018).

[6] A. Brack, A. Hoppe, M. Stocker, S. Auer, and R. Ewerth, "Analyzing the requirements for an open research knowledge graph: Use cases, quality requirements, and construction strategies"Int. J. on Digital Libraries **23** (1), 33–55 (2022). https://doi.org/10.1007/s00799-021-00306-x

[7] O. A. Nevzorova, N. Zhiltsov, A. Kirillovich, and E. Lipachev, "OntoMath$^{\mathrm{PRO}}$ ontology: A linked data hub for mathematics", Communications in Computer and Information Science **468**, 105–119 (2014). https://doi.org/10.1007/978-3-319-11716-4_9

[8] M. Kohlhase, *OMDoc – An Open Markup Format for Mathematical Documents [Version 1.2]* (Lecture Notes in Artificial Intelligence **4180**, Springer, Berlin–Heidelberg, 2006). https://doi.org/10.1007/11826095

[9] P.-O. Dehaye, M. Iancu, M. Kohlhase, A. Konovalov, S. Lelièvre, D. Müller, M. Pfeiffer, F. Rabe, N. M. Thiéry, and T. Wiesing, "Interoperability in the OpenDreamKit project: The Math-in-the-Middle approach", Intelligent Computer Mathematics (Lecture Notes in Computer Science **9791**, Springer, Cham, 2016), pp. 117–131. https://doi.org/10.1007/978-3-319-42547-4_9

[10] G. Gao, H. Ju, J. Jiang, Z. Qin, and B. Dong, "A semantic search engine for Mathlib4", arXiv preprint arXiv:2403.13310 (2024). https://arxiv.org/abs/2403.13310

[11] J. Asher, "LeanExplore: A search engine for Lean 4 declarations", arXiv preprint arXiv:2506.11085 (2025). https://arxiv.org/abs/2506.11085

[12] J. Lu, J. Liu, X. Wan, Y. Liu, and W. Lu, "Lean Finder: Semantic search for Mathlib that understands user intents", arXiv preprint arXiv:2510.15940 (2025). https://arxiv.org/abs/2510.15940

[13] Q. Chen, M. Yang, Z. Zhang, X. Zhang, Q. Zhang, J. Xu, X. Li, J. Zhang, X. Ma, X. Wang, and others, "AI4Research: A survey of artificial intelligence for scientific research", arXiv preprint arXiv:2507.01903 (2025). https://doi.org/10.48550/arXiv.2507.01903

[14] O. M. Ataeva and V. A. Serebryakov, "Ontology of a digital semantic library", Informatics and Applications **12** (1), 2–10 (2018). https://doi.org/10.14357/19922264180101

[15] Z. Shao, P. Wang, Q. Zhu, R. Xu, J. Song, and others, "DeepSeekMath: Pushing the limits of mathematical reasoning in open language models", arXiv preprint arXiv:2402.03300 (2024). https://doi.org/10.48550/arXiv.2402.03300

[16] H. Xin, Z. Z. Ren, J. Song, Z. Shao, and others, "DeepSeek-Prover-V1.5: Harnessing proof assistant feedback for reinforcement learning and Monte-Carlo tree search", arXiv preprint arXiv:2408.08152 (2024). https://doi.org/10.48550/arXiv.2408.08152

[17] DeepSeek-AI, "DeepSeek-Prover-V2: Advancing formal mathematical reasoning via reinforcement learning for subgoal decomposition", arXiv preprint arXiv:2504.21801 (2025). https://doi.org/10.48550/arXiv.2504.21801

[18] Numina and Kimi Team, "Kimina-Prover Preview: Towards large formal reasoning models with reinforcement learning", arXiv preprint arXiv:2504.11354 (2025). https://doi.org/10.48550/arXiv.2504.11354

[19] The Harmonic Team, "Aristotle: IMO-level automated theorem proving", Technical Report (2025).

[20] S. Varambally, T. Voice, Y. Sun, Z. Chen, R. Yu, and K. Ye, "HILBERT: Recursively building formal proofs with informal reasoning", in *NeurIPS 2025 Workshop on Mathematical Reasoning and AI (MATH-AI)*.

[21] G. Gao, Y. Wang, J. Jiang, Q. Gao, Z. Qin, T. Xu, and B. Dong, "Herald: A natural language annotated Lean 4 dataset", Int. Conf. on Learning Representations (ICLR 2025).

[22] R. Hsiang, W. Adkisson, R. J. George, and A. Anandkumar, "LeanDojo-v2: A comprehensive library for AI-assisted theorem proving in Lean", in *MATH-AI: The 5th Workshop on Mathematical Reasoning and AI at NeurIPS 2025* (San Diego, CA, USA, 2025).

[23] R. Hadsell, S. Chopra, and Y. LeCun, "Dimensionality reduction by learning an invariant mapping", 2006 IEEE Computer Society Conf. on Computer Vision and Pattern Recognition (CVPR 2006) **2**, 1735–1742 (2006). https://doi.org/10.1109/CVPR.2006.100

[24] A. van den Oord, Y. Li, and O. Vinyals, "Representation learning with contrastive predictive coding", arXiv preprint arXiv:1807.03748 (2018). https://doi.org/10.48550/arXiv.1807.03748

[25] T. Chen, S. Kornblith, M. Norouzi, and G. Hinton, "A simple framework for contrastive learning of visual representations", Proc. 37th Int. Conf. on Machine Learning (ICML 2020), Proc. Mach. Learn. Res. **119**, 1597–1607 (2020). https://proceedings.mlr.press/v119/chen20j.html

[26] A. Radford, J. W. Kim, C. Hallacy, A. Ramesh, G. Goh, S. Agarwal, G. Sastry, A. Askell, P. Mishkin, J. Clark, G. Krueger, and I. Sutskever, "Learning transferable visual models from natural language supervision", Proc. 38th Int. Conf. on Machine Learning (ICML 2021), Proc. Mach. Learn. Res. **139**, 8748–8763 (2021). https://proceedings.mlr.press/v139/radford21a.html

[27] Y. Tian, D. Krishnan, and P. Isola, "Contrastive multiview coding", in *Computer Vision – ECCV 2020*, Lecture Notes in Computer Science **12356**, 776–794 (Springer, Cham, 2020). https://doi.org/10.1007/978-3-030-58621-8_45

[28] T. R. Gruber, "A translation approach to portable ontology specifications", Knowledge Acquisition **5** (2), 199–220 (1993). https://doi.org/10.1006/knac.1993.1008

[29] A. Tarski, "The semantic conception of truth and the foundations of semantics", Philosophy and Phenomenological Research **4** (3), 341–376 (1944). https://doi.org/10.2307/2102968

[30] N. Gerasimenko, A. Vatolin, A. Ianina, and K. Vorontsov, "SciRus: Tiny and powerful multilingual encoder for scientific texts", Doklady Mathematics **110** (1), S193–S202 (2024). https://doi.org/10.1134/S1064562424602178

[31] K. Zheng, J. M. Han, and S. Polu, "miniF2F: A cross-system benchmark for formal Olympiad-level mathematics", arXiv preprint arXiv:2109.00110 (2021), published as a conference paper at ICLR 2022. https://doi.org/10.48550/arXiv.2109.00110

[32] F. Wilcoxon, “Individual comparisons by ranking methods”, Biometrics Bulletin **1** (6), 80–83 (1945). https://doi.org/10.2307/3001968