MATHATLAS: A BENCHMARK FOR AUTOFORMALIZATION *in the Wild*

Anonymous authors
Paper under double-blind review

ABSTRACT

Prior autoformalization benchmarks are largely focused on olympiad or undergraduate mathematics, while graduate and research-level mathematics remains under-studied. In this paper, we introduce *MathAtlas*, the first large-scale autoformalization benchmark of *in-the-wild* graduate-level mathematics, containing $\sim 52k$ theorems, definitions, exercises, examples, and proofs extracted from 103 graduate mathematics textbooks. *MathAtlas* is enriched with a mathematical dependency graph containing $\sim 178k$ relations, and is the first autoformalization benchmark to include such relations, facilitating evaluation and development of dependency-aware autoformalization systems. Our extensive experiments reveal several gaps in current autoformalization systems. First, we find naive, single-pass systems are insufficient for high-level mathematics. Second, we find feedback-aware formalization loops are an improvement, but primarily only help compilation rate, not semantic faithfulness. Third, we find frontier agentic systems, though effective, are extremely expensive and inefficient, thereby limiting at-scale autoformalization and making efficiency a key challenge. Furthermore, we find system performance degrades substantially with dependency depth. To study performance on "deep" mathematics, we release *MA-Hard*, a split of $\sim$700 entities with the deepest dependency trees. On *MA-Hard*, our strongest agentic system achieves 64% correctness, but does so inefficiently. We release *MathAtlas* to the community as a benchmark set for large-scale autoformalization of graduate-level mathematics *in the wild*. Finally, we introduce *MA-Align*, a semantic-faithfulness benchmark containing 200 labeled graduate-level examples covering both statements and definitions. Existing metrics perform significantly worse on *MA-Align* than on prior undergraduate-level benchmarks, indicating a need for further work in this area.

1 INTRODUCTION

The popularity of formal mathematics (computer verifiable mathematics) within the mathematics community has grown dramatically over recent years (Scholze & Buzzard, 2020; Tao, 2023; Kontorovich & Tao, 2024). Together with advances in AI, this has caused a surge of interest in **autoformalization** (Gadgil et al.; Wu et al.; Jiang et al., 2023; Azerbayev et al.; Liu et al., 2025a), the task of automatically converting natural language math to a formal verifiable language such as Lean (Moura & Ullrich, 2021).

Until recently, autoformalization work has been largely focused on simpler settings such as olympiad or undergraduate mathematics (Azerbayev et al.; Wu et al.; Zheng et al.; Exp; Agrawal et al.; Gao et al., 2025b; Liu et al., 2025b), where mathematical statements require limited prerequisite material to properly formalize, much of which already has been formalized by humans in libraries such as Mathlib (mathlib Community, 2020).

However, in more advanced mathematics, autoformalization becomes increasingly complex. Advanced mathematics has a large amount of prerequisite theory, much of which has yet to be formalized. This means an autoformalization system must first retrieve or produce the necessary dependent theory, correctly formalize it, and then formalize the original target statement.

Additionally, existing benchmarks typically focus on theorem statements, and ignore formalizations of mathematical *definitions*. Systems which can only autoformalize theorem statements are limited

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

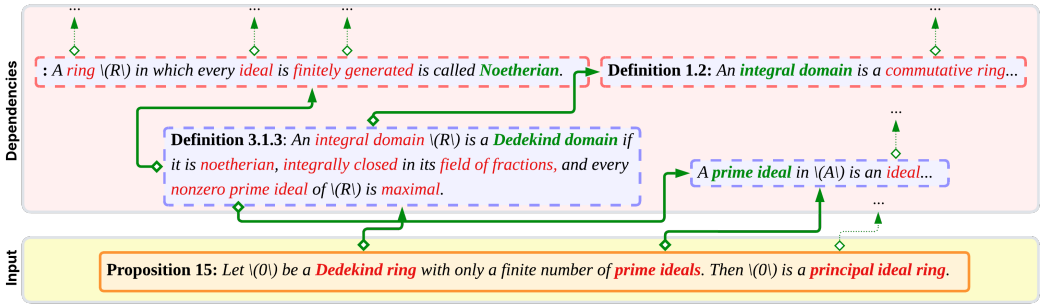


Figure 1: An entity from *MathAtlas* together with a subgraph of its dependency tree. In yellow (bottom), we have a target theorem statement to be formalized (Proposition 15). Proposition 15 contains three "object references" to previously defined mathematical concepts: Dedekind ring, prime ideals, and principal ideal ring, which are dependencies needed to correctly formalize the theorem. These objects each have their own dependency tree (shown with a red border), and so on. Formalizations of objects possibly already exist (e.g., in Mathlib), or they need to be synthesized before the target statement can be formalized.

in their scope by the level of existing theory because they cannot formalize theorems using math concepts they do not already have definitions for. Thus, creating scalable autoformalization systems necessitates the study of definition autoformalization, for which large-scale real-world benchmarks are lacking (Zhang et al., 2025a; Wang et al., 2025b).

To comprehensively evaluate the performance of autoformalization methods in higher-level mathematics, we present ***MathAtlas***,[1] a large-scale, graduate-level benchmark designed to measure autoformalization performance on mathematics in the wild. *MathAtlas* contains $\sim 52k$ informal entities, divided into statements (theorems, examples, and exercises), definitions, and proofs, which are extracted from 103 graduate math textbooks. Entities in *MathAtlas* span 87 areas of mathematics, including real analysis, differential geometry, category theory, topology, quantum groups, among others. Some entities in *MathAtlas* are formalizable on top of Mathlib, but many require additional formal theory which may not currently exist. To facilitate access to these required dependencies, we enrich each entity with long-range dependency relations, local variable references, and local context (see Figure 1).

MathAtlas is the first large-scale, *in the wild* autoformalization benchmark of graduate-level mathematics, the first to include rich contextual and dependency data, and the first to annotate their inter-entity dependencies. Our extensive experiments show that *MathAtlas* is very challenging. Furthermore, we find performance is correlated with an entity's *dependency depth*. To further study this difficulty, we construct ***MA-Hard***, a subset of 700 entities with the deepest dependency trees. Based on our experiments, we draw three main conclusions.

First, single-pass systems (especially open-source) are efficient but weak: the best fine-tuned and few-shot methods reach at most 9.8% correctness on statements and 16.7% on definitions. Second, compiler-feedback loops greatly improve correctness, but almost entirely by improving compilation rate, not faithfulness. Third, agentic systems such as Claude Code are highly effective, (64.1% on *MA-Hard*) but highly expensive (costing over $6\times$ more per item than a single-pass with the same base model) which limits scalability and accessibility.

Lastly, we investigate how well existing semantic faithfulness metrics for autoformalization generalize to *MathAtlas*. We present ***MA-Align***,[2] a binary classification benchmark of 200 entities from *MathAtlas*. Unlike prior benchmarks such as ConsistencyCheck (Chen et al., 2026) and CriticLean-Bench (Peng et al., 2025), *MA-Align* contains both labeled examples of theorem statements and definitions, and covers a wide variety of mathematics at the graduate level. Our experiments show that prior baselines have significant room for improvement on *MA-Align*.

[1] https://huggingface.co/datasets/MathAtlas/MathAtlas
[2] https://huggingface.co/datasets/MathAtlas/MA-Align

Definition 2.7.10 (Invariant bilinear form).: Let $V = V_{\bar{0}} \oplus V_{\bar{1}}$ be a superspace and carrying a representation of a Lie algebra $\mathfrak{g}$. A bilinear form $(\cdot|\cdot)$ is said *invariant* if, for any homogeneous element $X \in \mathfrak{g}$ and $v, w \in V$, $(Xv|w) + p(v, X)(v|Xw) = 0$. Moreover, if V is the representation space of the adjoint representation, then the bilinear form is said *adjoint* invariant.

Corollary 3.A.24: Let X be a compact Kahler manifold. Then the inclusion $i : (\mathcal{A}^*(X)^c, d) \subset (\mathcal{A}^*(X), d)$ is a dga-quasi-isomorphism.

Example 5.1.14.: Find two fundamental matrices for the linear homogeneous system in Example 5.1.11.

Figure 2: Several entities from *MathAtlas* of various types: (1) a definition entity which defines two terms: "invariant" and "adjoint invariant". (2) a theorem statement of a corollary from complex geometry. (3) an example from an ODE textbook.

2 MATHATLAS BENCHMARK

MathAtlas is constructed automatically from textbook documents. We discuss the entity and relation types in *MathAtlas*, and then discuss our entity and relation extraction pipeline. We also measure the quality of *MathAtlas* with a human evaluation.

2.1 STRUCTURE OF *MathAtlas*

MathAtlas is a graph of 52,052 formalizable text spans (entities) from 103 textbooks, linked by about 178k dependency relations.

Entities We use the term **entity** to mean a span of informal text which can be formalized, such as a theorem in a textbook. In *MathAtlas*, entities are either definitions, theorems, proofs, examples, or exercises (see Figure 2). Entities can, but don't always, have **identifiers**, such as "Theorem 4.3." These identifiers are sometimes referenced by other entities (see Figure 1). Importantly, entities sometimes contain multiple formalizable items. For example, in Figure 2, Definition 2.7.10 actually defines two objects, and so its formalization may require two definitions.

References Entities can contain **references** to other entities, dependent mathematical objects, or local variables which are not explicitly defined in the entity itself. We call these *entity references*, *object references*, and *local variable references*, respectively.

Relations We also include relations between entity references and the target entity, and object references and entities which define those objects. For example, an entity e_{met} may define a *metric space*, and contain an object reference r_{top} to a *topological space*, which is defined in an entity e_{top}. In this case, we add a relation $\langle e_{met}, e_{top} \rangle$ (see Figure 1). We define *dependency depth* to be the maximum height of an item's dependency tree, where nodes are *entities* (e.g., definitions or theorems) and edges are *object references*.

Statistics We now discuss some statistics of *MathAtlas*. Overall, *MathAtlas* contains 52,052 entities, which are comprised of $\sim$ 18k theorems, $\sim$ 10k exercises, $\sim$ 10k definitions, $\sim$ 10k proofs, and 5k examples.

About 90% of entities contain at least one object reference. In sum, there are $\sim$193k object references, for an average of 3.69 per entity. Of these, our relation extraction system found a matching target entity for 83.7%, for a total of $\sim$161k total object relations in the full graph. In addition, 20.5% of entities contain an entity reference, summing to $\sim 17k$ total entity references. Of these, over 99.8% were matched with a target entity.

Nodes have an average dependency depth of 30 (median of 17), but range anywhere from 0 to 80. We also find a stark difference in dependency depth based on the field of mathematics. For example, entities from textbooks on Lie algebra or Quantum groups had an average depth of $\sim$ 50, whereas those from books on set theory and logic had an average depth of $\sim$ 6. This is in-line with the intuition that certain fields of mathematics have more prerequisites, and thus deeper dependency trees.

Table 1: Intrinsic data quality analysis of *MathAtlas*.

Ref. type	F1	Recall	Precision
Entity ref.	94.7%	100%	90%
Object ref.	94.9%	98%	92%
Overall	94.8%	99%	91%

(a) Performance of our relation extraction system on a human-annotated subset of 100 extracted relations (50 of each type). To measure recall, we annotate gold target entities from among the retrieved candidates; if no selected candidate is an appropriate match, we assume none exists.

Entity Type	Valid %
Definition	98.0%
Theorem	94.0%
Proof	80.0%
Example	86.0%
Exercise	94.0%
Overall	90.4%

(b) Precision of our entity extraction system on a human-annotated subset of 250 examples (50 of each type).

2.2 Construction of *MathAtlas*

Here we give an overview of our dataset construction pipeline (see Appendix A for details).

We collect 103 graduate-level math textbooks spanning 87 fields and subfields, convert them to MMD with Nougat, and filter out non-content pages (indices, glossaries). We then run entity, reference, and (for definitions) name extractors, and build the dependency graph with a relation extractor (Appendix A).

Following our pipeline (Figure 4, Appendix), we convert the PDFs to MMD (mathematical markdown) files and filter out pages containing metadata, such as indices or glossaries. Then, we run an entity extraction model on the MMDs, which identifies spans of text corresponding to entity types, along with any identifier found in the entity (e.g., "Example 3.4"). Next, on any extracted entity, we run a reference extractor to identify any references to other entities, objects, or local variables. Lastly, on *definition* entities, we run a name extraction system, which identifies the name(s) of the defined object(s).

After extracting entities and associated metadata, we construct our dependency graph with a relation extraction system (see Appendix A for system details). We add "refers-to" relations between entity references and their target entities, and between an object reference and the entity which defines that object, if present. Finally, we add a "proves" relation between a proof entity and the theorem entity it proves.

2.3 Quality Analysis of *MathAtlas*

To evaluate entity extraction quality, we annotate a random sample of 250 extracted entities from *MathAtlas*, 50 entities of each type, based on whether the entity is valid.[3] For an entity to be valid, it must be complete (e.g., not cut off at the start or end, no missing segments due to improper PDF-markdown conversion), and have a correctly labeled type. We find that overall, 90.4% of the annotated entities are correctly extracted (see Table 1). Proofs are notably poorly extracted compared to others, primarily due to over-extraction (containing additional, unnecessary tokens).

We also annotate 100 randomly sampled references, comprised of 50 object references and 50 entity references. Like before, references are considered correct if they are complete and are labeled with the correct type. We find that 96% of references were properly extracted.

Finally, we measure the performance of our relation extraction system. For each of the 100 randomly sampled references above, we retrieve 10 candidate targets with our retriever and annotate which, if any, are correct. We compute precision, recall, and F1 score of our automated system against the human annotations. Overall, we find an F1 score of 94.8% (see Table 1a).

[3]Our annotators hold, at minimum, undergraduate math degrees.

Table 2: Faithfulness judges on *MA-Align* and on two external benchmarks, all run through the judging path used to score every system. We evaluate models on balanced accuracy, and report a 95% confidence interval by bootstrap resampling. We report balanced accuracy to account for class imbalance in the dataset. We also report scores for gpt-5.2 and Claude Sonnet for comparison purposes only.

	MA-Align Defs.	*MA-Align* Stmts.	External (Bal.)	
Judge	Bal.	Bal.	ConsistencyCheck	CriticLeanBench
CriticLean-32B	65.1 ± 9.8	78.5 ± 7.3	77.4	82.0
gpt-5.2	72.0 ± 8.6	77.7 ± 9.0	65.6	79.0
Claude Sonnet	83.1 ± 5.5	78.6 ± 7.9	85.5	90.1
Qwen3.8 27B	77.3 ± 8.5	77.8 ± 6.9	83.6	87.0

3 AUTOFORMALIZATION METRICS FOR *MathAtlas*

Following previous work, we use the standard autoformalization metric of **compile rate**.[4] However, compile rate does not measure faithfulness of the formalization, so we also report *semantic faithfulness* (see below). Finally, we report a combined metric of compile rate and semantic faithfulness which we call **correctness**. A formal statement must both compile and be faithful to the informal statement for it to be considered correct.

Semantic Faithfulness Metric Together with syntactic correctness (compilation rate), *semantic faithfulness* is a critical metric for autoformalization. Faithfulness measures whether a formal statement has the same mathematical meaning as its informal source. Prior work on semantic faithfulness (Lu et al., 2024; Chen et al., 2026; Peng et al., 2025; Liu et al., 2026) focuses on easier domains (olympiad or undergraduate math), and does not measure alignment of definitions. *MathAtlas* highlights this gap in prior work, as it contains both high-level mathematics and definition entities.

To assess the quality of semantic faithfulness metrics on graduate-level, *in the wild* mathematics, we introduce *MA-Align*, a semantic faithfulness benchmark of 200 syntactically correct statements and definitions from *MathAtlas*. Each informal entity is formalized by `gpt-oss-120b`, and annotated with a binary label, "aligned" or "misaligned".[5] We evaluate prior LLM-as-judge systems, CriticLean (Peng et al., 2025), as well as our few-shot LLM-as-judge on ConsistencyCheck (Chen et al., 2026) and CriticLeanBench (Peng et al., 2025), and *MA-Align* (see Table 2). [6]

Semantic Faithfulness Experimental Results While Claude Sonnet has the highest overall performance, we find that Qwen 3.8 27B is also strong. We find that CriticLean performs adequately on statements, but particularly poorly on *MA-Align*-defs, making it not an ideal choice for the *MathAtlas* domain.

Despite stronger performance, Claude Sonnet is both proprietary (thus might change or be deprecated without warning) and expensive to use. Thus we elect to use Qwen 3.8 27B as our primary semantic faithfulness metric for *MathAtlas*, as it is still well-correlated with human evaluations, is open source, and more accessible. We note that the primary error of Qwen 3.8 over-acceptance of incorrect formalizations, meaning reported correctness scores are likely slightly inflated rather than under-sold.

4 AUTOFORMALIZATION EXPERIMENTS

In this section, we present autoformalization experiments on *MathAtlas* with several strong baseline methods. We first detail the various systems and settings, followed by an analysis of the results.

[4] We use the software package `blv` (Nilay Patel, 2026) to measure compile rate, compiled against Lean version v4.24.0.

[5] We selectively sample 200 such entities to better balance the label distribution.

[6] We note that we leave out FormalAlign (Lu et al., 2024) as a baseline as there is no official implementation, and our reimplementation was unable to match the reported results.

4.1 Dataset Splits

We split *MathAtlas* into a "statement" split, comprised of all exercises, examples, and theorems (as they are all formalized similarly), and a "definition" split. Due to budgetary constraints, we evaluate statement formalization only and leave proofs to future work.

Open Split Approximately 40% of entities in *MathAtlas* are extracted from Springer textbooks. To comply with copyright law, we release code to generate *MathAtlas* in full, which can be run by those who have access to Springer textbooks. However, to facilitate access to *MathAtlas*, we release an open split of the remaining 60% of the data publicly.[7] Importantly, the distribution of fields, dependency depth, and system performance is comparable between the Open Split and the full dataset. Our results on this open subset are reported in the Appendix in Table 6.

MA-Hard We select a subset of ~ 700 examples with the deepest dependency trees, which we call *MA-Hard*. This set primarily consists of examples from more advanced graduate-level mathematics, such as Algebraic Geometry and Lie Theory, and spans 14 subjects. We also note that *MA-Hard* consists of both statements (~ 600) and definitions (~ 100). All examples from *MA-Hard* are in the open split.

4.2 Evaluated Methods

We measure the performance variety of autoformalization baselines, including fine-tuned, few-shot, and agentic methods. Due to compute and price limitations, we evaluate the more expensive systems only on *MA-Hard*.

For few-shot prompting, we evaluate a strong open-source model, `gpt-oss-120B`, and a frontier API model, Claude Sonnet. We ablate the inclusion of each item's dependencies (as in *MathAtlas*), and a feedback-aware iteration loop. We supply the target entity text as input, and the expected output is a syntactically and semantically correct formal representation in Lean 4 (see § 3). This approach has been extensively studied, especially on benchmarks such as miniF2F (Zheng et al.) and ProofNet (Azerbayev et al.). We also do a prompt ablation, evaluating the inclusion of out-of-domain few-shot examples, an engineered prompt, in-domain few-shot examples, and local-context.

For our agent baseline, we supply the agent with the target text and ID, and supply it with web search, access to a Lean compiler (Dressler, 2025), and LeanSearch (Gao et al., 2026) for searching mathlib.

On statements, we also test fine-tuned methods from prior work: Herald Translator (Gao et al., 2025b), Goedel Formalizer v2 (Lin et al., 2025), Kimina Autoformalizer (Wang et al., 2025a), Atlas Translator (Liu et al., 2025b), and ReForm (Chen et al., 2026). However, for definitions, we do not include any fine-tuned methods, as our experiments revealed autoformalization methods for theorems were unable to generalize to definitions in this domain.

Local Context As discussed in subsection 2.1, some entities in *MathAtlas* require context to define variables, entities, or otherwise elaborate a mathematical setting. We evaluate a simple baseline of including the prior 500 tokens in the context window of the LLM. For this setting, we test only the few-shot prompted systems, as this is out-of-domain for the fine-tuned models. In addition, we provide a tool for agents to access local context from textbooks. In this way, we allow the systems to get context only when necessary, mitigating noise.

Feedback Iteration A common approach to improving autoformalization is providing compiler feedback to a system after an initial attempt, and letting it repair the statement based on errors. We test this approach on our few-shot systems, allowing $k = 5$ rounds of iteration. Agents are provided via lean-lsp-mcp (Dressler, 2025) to iterate on feedback.

4.3 Results

Full Dataset: Single-Pass Systems Overall, we find *MathAtlas* to be very challenging, in particular for open-source models. On the *statement* split, our strongest baseline is ReForm 8B, which achieved an overall correctness of 9.8% (see Table 3). We note the disparity between compilation

[7] `https://huggingface.co/datasets/MathAtlas/MathAtlas`

Table 3: Results for autoformalization experiments on *MathAtlas*. "Compiles" measures the percentage of examples which successfully compile. "Faithful" measures what percent of the compiling theorems are faithful. "Correct" indicates the overall percentage of statements which compile and are faithful.

Model	Comp.	Faith.	Cor.
Prompted			
gpt-oss-20b (0-shot)	10.5%	20.9%	2.2%
+few-shot	18.1%	41.4%	7.5%
+tuned prompt	17.1%	47.4%	8.1%
+tuned exs.	20.8%	56.3%	11.7%
+context	16.1%	55.9%	9.0%
gpt-oss-120b (0-shot)	13.4%	49.3%	6.6%
+few-shot	24.2%	57.0%	13.8%
+tuned prompt	26.0%	58.8%	15.3%
+tuned exs.	**28.5%**	58.5%	**16.7%**
+local context	25.8%	**58.9%**	15.2%

(a) Autoformalization results for definitions. We do not evaluate fine-tuned methods for definitions because there are no prior-work models available for definitions, and "theorem" models do not generalize. The best performance in each category and column are bolded.

Model	Comp.	Faith.	Cor.
Fine-tuned			
Herald 7B	11.8%	12.7%	1.5%
Kimina 7B	**27.3%**	8.4%	2.3%
ATLAS-L 8B	21.4%	16.4%	3.5%
Goedel 8B	20.3%	34.9%	7.1%
Goedel 32B	23.2%	30.6%	7.1%
ReForm 8B	19.0%	**48.4%**	**9.8%**
Prompted			
gpt-oss-20b (zs)	9.9%	42.4%	4.2%
+few-shot	16.5%	35.8%	5.9%
+tuned prompt	16.8%	32.7%	5.5%
+tuned exs.	19.1%	35.1%	6.7%
+local context	18.6%	36.0%	6.7%
gpt-oss-120b (zs)	14.3%	**47.6%**	6.8%
+few-shot	16.8%	43.5%	7.3%
+tuned prompt	17.1%	43.9%	7.5%
+tuned exs.	**21.7%**	35.9%	**7.8%**
+local context	20.0%	34.5%	6.9%

(b) Autoformalization results for statements (theorems, examples, and exercises). The best performance in each category and column are bolded.

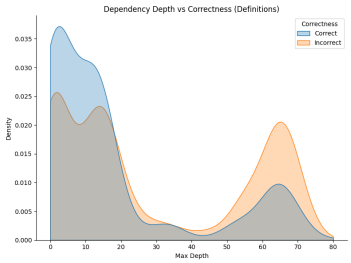
rate and correctness, especially in fine-tuned models: Kimina 7B, for example, achieves 27.3% compilation rate, but just 2.3% of formalizations compile and are faithful. On the *definition* split, our strongest system is `gpt-oss-120b`, with 16.7% correctness, likewise with a significant 11.8% gap between compilation rate and overall correctness.

Local Context Many items in *MathAtlas* require local context to correctly solve. However, including local context seems to confuse prompted systems, and the input decreases performance of the best model from 16.7% to 15.2% on definitions, and 7.8% to 6.9% on statements (see Table 3). This drop in performance indicates further research is necessary on incorporating local context into autoformalization methods.

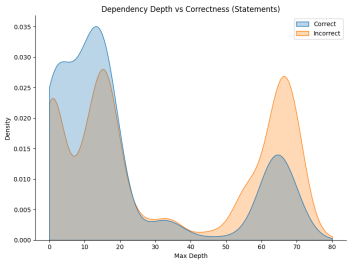
Object Dependencies Next, we compare correctness versus max dependency depth. In Figure 3, we notice a clear trend, showing that our best systems is less likely to correctly formalize deeper targets. We compute a two-sample Kolmogorov–Smirnov, and find significant difference between the max depth distributions for correct and incorrect examples ($p < 0.001$).

Existence in Mathlib We compare correctness of definitions which are and aren't grounded in Mathlib (see Figure 10). We find that 27.9% of definitions which our linker grounded in Mathlib are correctly formalized, whereas only 16.8% of missing definitions are correct, statistically significant difference (chi-square $p < 0.001$). A possible explanation for this is that Mathlib is likely in the training data for evaluated LLMs. Additionally, since mathlib is formalized "bottom-up", it is less likely to have highly advanced, complex entities, which tend to be harder to formalize.

***MA-Hard*: Feedback-Aware Systems** On *MA-Hard*, single-pass systems perform poorly, in particular open-source ones (see Table 4): the best fine-tuned model, ReForm 8B, achieves 1.9% correctness, and `gpt-oss-120b` 2.7%. Adding dependencies to the context does not meaningfully help these systems. Feedback-aware iteration ($k = 5$), however, improves compilation rate substantially: from 8.6% to 34.7% for `gpt-oss-120b`, and from 55.2% to 99.7% for Claude Sonnet. This does not carry over to faithfulness, which stays flat for Sonnet (50.0% to 50.9%) and drops for `gpt-oss-120b` (31.7% to 22.7%). Iteration therefore raises correctness (to 7.9% and 50.7%, respectively) mainly by making more outputs compile, not by making them more faithful.



(a) Correctness versus dependency depth of our best performing single-pass system for definitions (`gpt-oss-120b`).



(b) Correctness versus dependency depth of our best performing single-pass system for statements (ReForm 8B)

Figure 3: Correctness of our strongest single-pass models compared with dependency depth on examples in *MathAtlas*.

Table 4: Performance of baseline systems on *MA-Hard*.

Model / setting	Compiles	Faithful	Correct
Fine-tuned methods (statements-only)			
ReForm 8B	5.0%	38.7%	**1.9%**
Goedel 8B	6.6%	19.5%	1.3%
Goedel 32B	8.2%	19.6%	1.6%
Kimina 7B	12.2%	2.6%	0.3%
Herald 7B	3.4%	23.8%	0.8%
Few-shot			
gpt-oss-120b	8.6%	31.7%	2.7%
+ deps in context	7.2%	36.0%	2.6%
+ iterative ($k = 5$)	34.7%	22.7%	7.9%
gpt-5.2	12.8%	24.7%	3.2%
+ deps in context	17.8%	25.0%	4.4%
Claude Sonnet	55.2%	50.0%	27.6%
+ deps in context	54.4%	54.2%	29.5%
+ iterative ($k = 5$)	99.7%	50.9%	**50.7%**
Agentic			
Claude Code (Sonnet)	95.8%	59.9%	57.4%
+dependency-aware	97.3%	65.8%	**64.1%**

***MA-Hard*: Agentic Systems** Agentic systems are the strongest we evaluate. Claude Code with Sonnet reaches 57.4% correctness, with a faithfulness rate of 59.9% among compiling outputs, higher than any single-pass or iterative system. Access to *MathAtlas*'s dependency graph improves this further, to 64.1% correctness, with the gain coming mostly from faithfulness (59.0% to 65.8%). In contrast, adding dependencies to the context of single-pass Sonnet gives only a small improvement (27.6% to 29.5%), suggesting that agents are better able to make use of dependency information when they can request it selectively.

Efficiency The strongest systems are also the least efficient (see Table 5). Single-pass systems are cheap, at 14k tokens and $0.031 per item for Sonnet, but underperform. Iterative on feedback improves compilation rate, but increases cost by approximately 2-3$\times$ for $k = 5$ loops. Agents are strong but expensive: Claude Code uses 5.4 calls and 164.6k tokens per item, and the dependency-aware variant 6.9 calls and 235.0k tokens ($0.191 per item, $133 in total), roughly six times the cost of single-pass Sonnet for a 2.3$\times$ gain in correctness. Other, more complex systems can be even more expensive. ARIA autoformalizer (Wang et al., 2025b), for example, took over 15 minutes per problem in our tests, and over $3 per problem, prohibiting testing ARIA on *MA-Hard*. These costs limit the practicality of running complex agentic systems on larger datasets such as *MathAtlas*. Improving efficiency is therefore a key open problem in scaling up autoformalization.

5 Related Works

LLMs for Autoformalization Large language models (LLMs) have made significant advancements in autoformalization. Early work with LLMs such as in Wu et al. used naive approaches, but showed promise in the methodology. Further refinements with fine-tuning on various data made significant progress (Gao et al., 2025b; Lin et al., 2025; Wu et al., 2025; Liu et al., 2025b; Azerbayev et al., 2023) but, as we show in this paper, face similar issues in more difficult settings. More recently, there has been work on breaking down targets either by decomposing into subproblems (Shi et al., 2026) or retrieving dependencies first (Wang et al., 2025b; Min et al., 2026). These methods have encouraging results in harder settings (Jiang et al., 2025), but use proprietary and expensive models and are token-inefficient. ProofFlow (Cabral et al., 2025) introduces a method for formalizing natural language proofs, as well as ProofScore, a metric for measuring faithfulness to the natural language.

Autoformalization Benchmarks Prior autoformalization benchmarks have largely focused on formalizing theorem statements (Zheng et al.; Azerbayev et al.), primarily on olympiad or undergraduate mathematics. LeanEuclid (Murphy et al., 2024) is a benchmark for autoformalizing geometry from Euclid's *Elements* and the UniGeo dataset (Chen et al., 2022). Zhang et al. (2025a) present two definition-specific benchmark of 100 definitions taken from arXiv and Wikipedia, and show proprietary LLMs have low performance. The FATE-H and FATE-X benchmarks (Jiang et al., 2025) consist of 100 problems from commutative and abstract algebra, and pose a significant challenge compared to previous work. In contrast to the FATE benchmarks, *MathAtlas* is much larger, contains multiple entity types, is more varied in topic, and includes local and long-range dependencies which facilitate the study of isolated system components, such as retrieval or synthesis.

Autoformalization Metrics Correctness in autoformalization is measured with two metrics: compilation and semantic faithfulness. The former is simple to measure with the language's compiler, or specialized tools such as Kimina Lean Server (Santos et al., 2025) or blv (Nilay Patel, 2026).

More difficult to measure is faithfulness, which measures whether the generated formal language code is semantically equivalent to the input informal statement. FormalAlign (Lu et al., 2024) introduces a methodology for fine-tuning an alignment model for statements. However, the methodology does not scale well to more difficult domains, and fails to capture the types of mistakes made by more advanced autoformalization models.[8] For benchmarks with gold formal labels, BEq (Liu et al., 2025a) and BEq+ Poiroux et al. (2025) judges semantic equivalence between two formal statements based on definitional equivalence, and TransTED (Liu et al., 2026) presents a tree-based similarity metric. ProofScore (Cabral et al., 2025) as previously mentioned is a faithfulness metric specifically for proofs. Using an LLM-as-judge approach has been common for measuring semantic faithfulness (Zhang et al., 2025b), and has shown promise as a scalable metric. ConsistencyCheck (Chen et al., 2026) is a benchmark for semantic faithfulness metrics, but as we show in § 3, high performance on ConsistencyCheck does not necessarily generalize to harder domains, e.g., *MA-Align*.

6 Conclusion

We introduced *MathAtlas*, a large-scale benchmark for graduate-level autoformalization in the wild, with a dependency graph. Our experiments show that performance degrades with dependency depth. Single-pass systems, while efficient, have poor performance overall. Feedback iteration improves compilation but not faithfulness, and agentic systems are strongest (64% on *MA-Hard*) but costly, making efficiency a key open problem. Our *MA-Align* benchmark shows that existing faithfulness metrics, including CriticLean and strong LLM judges, are effective at measuring faithfulness, but have clear room for improvement. We release *MathAtlas*, *MA-Align*, and our code.

[8] No implementation or model weights have been released as of this paper. Our analysis of these results is based on our re-implementation, which failed to match reported numbers.

AI USE STATEMENT

AI tools were used in this paper as part of the dataset construction pipeline. Details are thoroughly discussed in the main paper, with further details in the appendix. Furthermore, we use AI tools to aid in finding related works. Finally, we use AI spell-check and grammar-check tools. We have implemented guardails for any AI extracted data to ensure no hallucinations, and have verified AI suggested retrievals.

REFERENCES

Exploration of neural machine translation in autoformalization of mathematics in Mizar — Proceedings of the 9th ACM SIGPLAN International Conference on Certified Programs and Proofs. URL https://dl.acm.org/doi/10.1145/3372885.3373827.

Ayush Agrawal, Siddhartha Gadgil, Navin Goyal, Ashvni Narayanan, and Anand Tadipatri. Towards a Mathematics Formalisation Assistant using Large Language Models. URL http://arxiv.org/abs/2211.07524.

Zhangir Azerbayev, Bartosz Piotrowski, and Jeremy Avigad. ProofNet: A Benchmark for Autoformalizing and Formally Proving Undergraduate-Level Mathematics Problems.

Zhangir Azerbayev, Hailey Schoelkopf, Keiran Paster, Marco Dos Santos, Stephen McAleer, Albert Q. Jiang, Jia Deng, Stella Biderman, and Sean Welleck. Llemma: An Open Language Model For Mathematics, October 2023.

Lukas Blecher, Guillem Cucurull, Thomas Scialom, and Robert Stojnic. Nougat: Neural optical understanding for academic documents, 2023.

Rafael Cabral, Tuan Manh Do, Xuejun Yu, Wai Ming Tai, Zijin Feng, and Xin Shen. ProofFlow: A Dependency Graph Approach to Faithful Proof Autoformalization, October 2025. URL http://arxiv.org/abs/2510.15981. arXiv:2510.15981 [cs].

Guoxin Chen, Jing Wu, Xinjie Chen, Wayne Xin Zhao, Ruihua Song, Chengxi Li, Kai Fan, Dayiheng Liu, and Minpeng Liao. Reform: Reflective autoformalization with prospective bounded sequence optimization, 2026. URL https://arxiv.org/abs/2510.24592.

Jiaqi Chen, Tong Li, Jinghui Qin, Pan Lu, Liang Lin, Chongyu Chen, and Xiaodan Liang. Unigeo: Unifying geometry logical reasoning via reformulating mathematical expression, 2022. URL https://arxiv.org/abs/2212.02746.

Oliver Dressler. Lean LSP MCP: Tools for agentic interaction with the Lean theorem prover, 3 2025. URL https://github.com/oOo0oOo/lean-lsp-mcp.

Robert Escriva, Hammad Bashir, Max Isom, Jeff Huber, Macronova, Sanket Kedia, itaismith, Luke VanderHart, Jai Radhakrishnan, tanujnay112, Trayan Azarov, Ben Eggers, Kyle Diaz, Liquan Pei, jasonvigil, Drew Kim, Anton Troynikov, Philip I. Thomas, Rohit P, nicolasgere, Gabe Shahbazian, Evan Culver, Cooper Gamble, Dave Dash, TJ Krusinski, Weili Gu, swyx.io, Matthew Keller, Really Him, and naynaly10. chroma-core/chroma. https://github.com/chroma-core/chroma, may 7 2026. URL https://github.com/chroma-core/chroma.

Siddhartha Gadgil, Anand Rao Tadipatri, Ayush Agrawal, Ashvni Narayanan, and Navin Goyal. Towards automating formalisation of theorem statements using large language models.

Guoxiong Gao, Haocheng Ju, Jiedong Jiang, Zihan Qin, and Bin Dong. A semantic search engine for mathlib4, 2025a. URL https://arxiv.org/abs/2403.13310.

Guoxiong Gao, Yutong Wang, Jiedong Jiang, Qi Gao, Zihan Qin, Tianyi Xu, and Bin Dong. Herald: A natural language annotated lean 4 dataset, 2025b. URL https://arxiv.org/abs/2410.10878.

Guoxiong Gao, Zeming Sun, Jiedong Jiang, Yutong Wang, Jingda Xu, Peihao Wu, Bryan Dai, and Bin Dong. Leansearch v2: Global premise retrieval for lean 4 theorem proving, 2026. URL https://arxiv.org/abs/2605.13137.

Albert Q. Jiang, Wenda Li, and Mateja Jamnik. Multilingual mathematical autoformalization. *arXiv preprint arXiv: 2311.03755*, 2023.

Jiedong Jiang, Wanyi He, Yuefeng Wang, Guoxiong Gao, Yongle Hu, Jingting Wang, Nailing Guan, Peihao Wu, Chunbo Dai, Liang Xiao, and Bin Dong. Fate: A formal benchmark series for frontier algebra of multiple difficulty levels, 2025. URL https://arxiv.org/abs/2511.02872.

Alex Kontorovich and Terence Tao. Prime Number Theorem and More, January 2024. URL https://github.com/AlexKontorovich/PrimeNumberTheoremAnd.

Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph E. Gonzalez, Hao Zhang, and Ion Stoica. Efficient memory management for large language model serving with pagedattention. In *Proceedings of the ACM SIGOPS 29th Symposium on Operating Systems Principles*, 2023.

Yong Lin, Shange Tang, Bohan Lyu, Ziran Yang, Jui-Hui Chung, Haoyu Zhao, Lai Jiang, Yihan Geng, Jiawei Ge, Jingruo Sun, Jiayun Wu, Jiri Gesi, Ximing Lu, David Acuna, Kaiyu Yang, Hongzhou Lin, Yejin Choi, Danqi Chen, Sanjeev Arora, and Chi Jin. Goedel-prover-v2: Scaling formal theorem proving with scaffolded data synthesis and self-correction, 2025. URL https://arxiv.org/abs/2508.03613.

Qi Liu, Xinhao Zheng, Xudong Lu, Qinxiang Cao, and Junchi Yan. Rethinking and improving autoformalization: towards a faithful metric and a dependency retrieval-based approach. In *The Thirteenth International Conference on Learning Representations*, 2025a. URL https://openreview.net/forum?id=hUb2At2DsQ.

Xiaoyang Liu, Kangjie Bao, Jiashuo Zhang, Yunqi Liu, Yu Chen, Yuntian Liu, Yang Jiao, and Tao Luo. Atlas: Autoformalizing theorems through lifting, augmentation, and synthesis of data, 2025b. URL https://arxiv.org/abs/2502.05567.

Xiaoyang Liu, Tao Zhu, Zineng Dong, Yuntian Liu, Qingfeng Guo, Zhaoxuan Liu, Yu Chen, and Tao Luo. Assess: A semantic and structural evaluation framework for statement similarity, 2026. URL https://arxiv.org/abs/2509.22246.

Jianqiao Lu, Yingjia Wan, Yinya Huang, Jing Xiong, Zhengying Liu, and Zhijiang Guo. FormalAlign: Automated Alignment Evaluation for Autoformalization, October 2024. URL http://arxiv.org/abs/2410.10135. arXiv:2410.10135 [cs].

The mathlib Community. The lean mathematical library. In *Proceedings of the 9th ACM SIGPLAN International Conference on Certified Programs and Proofs*, POPL '20. ACM, January 2020. doi: 10.1145/3372885.3373824. URL http://dx.doi.org/10.1145/3372885.3373824.

Marcus J. Min, Yeqi Gao, Wilson Sy, Zhaoyu Li, Xujie Si, and Osbert Bastani. Divide and abstract: Autoformalization via decomposition and abstraction learning. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=NjgaeXNit3.

Leonardo de Moura and Sebastian Ullrich. The lean 4 theorem prover and programming language, 2021. URL https://doi.org/10.1007/978-3-030-79876-5_37.

Logan Murphy, Kaiyu Yang, Jialiang Sun, Zhaoyu Li, Anima Anandkumar, and Xujie Si. Autoformalizing Euclidean geometry. In *International Conference on Machine Learning (ICML)*, 2024.

Nilay Patel. offendo/blv. https://github.com/offendo/blv, mar 11 2026. URL https://github.com/offendo/blv.

Zhongyuan Peng, Yifan Yao, Kaijing Ma, Shuyue Guo, Yizhe Li, Yichi Zhang, Chenchen Zhang, Yifan Zhang, Zhouliang Yu, Luming Li, Minghao Liu, Yihang Xia, Jiawei Shen, Yuchen Wu, Yixin Cao, Zhaoxiang Zhang, Wenhao Huang, Jiaheng Liu, and Ge Zhang. Criticlean: Critic-guided reinforcement learning for mathematical formalization, 2025. URL https://arxiv.org/abs/2507.06181.

Auguste Poiroux, Gail Weiss, Viktor Kunčak, and Antoine Bosselut. Reliable evaluation and benchmarks for statement autoformalization, 2025. URL https://arxiv.org/abs/2406.07222.

Marco Dos Santos, Haiming Wang, Hugues de Saxcé, Ran Wang, Mantas Baksys, Mert Unsal, Junqi Liu, Zhengying Liu, and Jia Li. Kimina lean server: Technical report, 2025. URL https://arxiv.org/abs/2504.21230.

Peter Scholze and Kevin Buzzard. Liquid tensor experiment, December 2020. URL https://xenaproject.wordpress.com/2020/12/05/liquid-tensor-experiment/.

Jiangze Shi, Zhiwei Zhang, Baoquan Ma, Shuai Zhao, Ye Yuan, and Guoren Wang. Loc-decomp: LLM autoformalization via logical concept decomposition and iterative feedback correction. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=0KFQ4F9YEH.

Terrance Tao. Formalizing the proof of pfr in lean4 using blueprint: a short tour, November 2023. URL https://terrytao.wordpress.com/2023/11/18/formalizing-the-proof-of-pfr-in-lean4-using-blueprint-a-short-tour.

Haiming Wang, Mert Unsal, Xiaohan Lin, Mantas Baksys, Junqi Liu, Marco Dos Santos, Flood Sung, Marina Vinyes, Zhenzhe Ying, Zekai Zhu, Jianqiao Lu, Hugues de Saxcé, Bolton Bailey, Chendong Song, Chenjun Xiao, Dehao Zhang, Ebony Zhang, Frederick Pu, Han Zhu, Jiawei Liu, Jonas Bayer, Julien Michel, Longhui Yu, Léo Dreyfus-Schmidt, Lewis Tunstall, Luigi Pagani, Moreira Machado, Pauline Bourigault, Ran Wang, Stanislas Polu, Thibaut Barroyer, Wen-Ding Li, Yazhe Niu, Yann Fleureau, Yangyang Hu, Zhouliang Yu, Zihan Wang, Zhilin Yang, Zhengying Liu, and Jia Li. Kimina-prover preview: Towards large formal reasoning models with reinforcement learning, 2025a. URL https://arxiv.org/abs/2504.11354.

Hanyu Wang, Ruohan Xie, Yutong Wang, Guoxiong Gao, Xintao Yu, and Bin Dong. Aria: An agent for retrieval and iterative auto-formalization via dependency graph, 2025b. URL https://arxiv.org/abs/2510.04520.

Liang Wang, Nan Yang, Xiaolong Huang, Linjun Yang, Rangan Majumder, and Furu Wei. Improving text embeddings with large language models, 2024. URL https://arxiv.org/abs/2401.00368.

Yuhuai Wu, Albert Q. Jiang, Wenda Li, Markus N. Rabe, Charles Staats, Mateja Jamnik, and Christian Szegedy. Autoformalization with Large Language Models. URL http://arxiv.org/abs/2205.12615.

Yutong Wu, Di Huang, Ruosi Wan, Yue Peng, Shijie Shang, Chenrui Cao, Lei Qi, Rui Zhang, Zidong Du, Jie Yan, and Xing Hu. Stepfun-formalizer: Unlocking the autoformalization potential of llms through knowledge-reasoning fusion, 2025. URL https://arxiv.org/abs/2508.04440.

Lan Zhang, Marco Valentino, and Andre Freitas. Autoformalization in the wild: Assessing LLMs on real-world mathematical definitions. In Christos Christodoulopoulos, Tanmoy Chakraborty, Carolyn Rose, and Violet Peng (eds.), *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pp. 1720–1738, Suzhou, China, November 2025a. Association for Computational Linguistics. ISBN 979-8-89176-332-6. doi: 10.18653/v1/2025.emnlp-main.90. URL https://aclanthology.org/2025.emnlp-main.90/.

Lan Zhang, Marco Valentino, and Andre Freitas. Beyond Gold Standards: Epistemic Ensemble of LLM Judges for Formal Mathematical Reasoning, June 2025b.

Kunhao Zheng, Jesse Michael Han, and Stanislas Polu. MiniF2F: A cross-system benchmark for formal Olympiad-level mathematics. URL https://arxiv.org/abs/2109.00110v2.

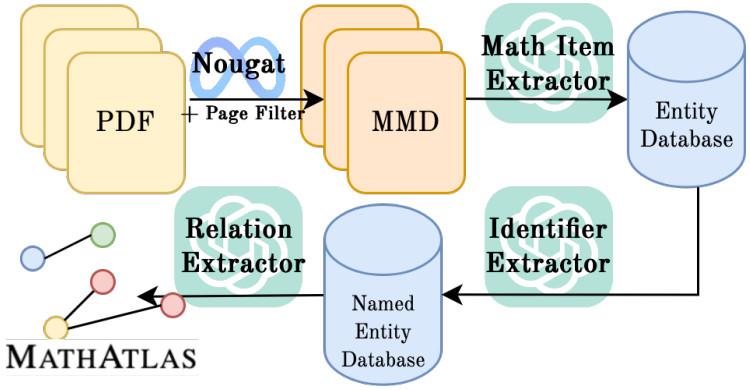


Figure 4: *MathAtlas* creation pipeline. For visual convenience, we combine the "name" and "reference" extractors into one step called "identifier extractor."

```
Text:
"M is compact iff G is abelian."

Reasoning:
- M not introduced → include local_variable_reference
- G not introduced → include local_variable_reference
- compact is a named property → include object_reference
- abelian is a named property → include object_reference

Output:
[
  {"term": "M", "reference_type": "local_variable_reference"},
  {"term": "G", "reference_type": "local_variable_reference"},
  {"term": "compact", "reference_type": "object_reference"},
  {"term": "abelian", "reference_type": "object_reference"}
]
```

Figure 5: Example of reference extraction system input and output.

A Extraction Pipeline

Our extraction pipeline starts with 103 selected textbooks covering a wide range of mathematics. We use Nougat (Blecher et al., 2023) to convert PDFs into Mathematical Markdown (MMD). We then run our entity extraction system, followed by our reference extraction and name extraction systems, and finally our relation extraction system. We detail each below.

Entity Extraction Our entity extraction system is a few-shot, prompt-engineered `gpt-oss-120b`. We prompt the model to generate an array of JSON objects with keys "type" indicating the entity's type, optionally an "identifier", which are spans such as Definition 3.9 or Theorem 15, and "text", which is the actual text of the entity. We enforce this structure with a JSON schema using structured decoding, a temperature of 0.3, and otherwise default sampling parameters.

We then run our reference extractor system, likewise a prompted *gpt-oss-120b* model. The reference extractor takes as input an extracted entity, and outputs a list of JSON objects with keys "term," indicating the text of the reference, and "reference_type," which is one of `object_reference`, `entity_reference`, or `local_variable_reference`. This is likewise enforced with JSON schema, and uses a temperature of 0.3.

Then, we run the name extractor, also a prompted `gpt-oss-120b`, prompted to output the names of objects defined in the entity. The output is a JSON array of strings, again enforced with a schema and structured decoding. We likewise use a temperature of 0.3.

Relation Extraction Each entity is embedded into a ChromaDB instance (Escriva et al., 2026) together with all its metadata, using the Mistral E5 model from Wang et al. (2024). Then, our relation extraction system works in two stages. First, we use the same model to retrieve candidate matches from the entity database. Secondly, we prompt `gpt-oss-120b` to determine which, if any, is the correct match, accounting for usage and context. If there are multiple matches, we prioritize those from the same textbook, or otherwise those with the highest matching score.

Grounding in Mathlib We use LeanSearch (Gao et al., 2025a) to link definitions in *MathAtlas* to formal definitions in Mathlib. We query the definition text to get 10 candidate matches, and select from the matches using a prompted "gpt-oss-120b." Unlike the authors, however, we find that augmenting the query degrades performance, so we skip that step in our final run. We suspect this is because our queries are already fully-formed definitions rather than keywords or phrases. We include the selected matches as well as all 10 candidates for each definition in the released dataset.

B PROMPTS

```
   ## Task
You are given a block of markdown mathematical text.
Your task is to identify and extract all contiguous snippets that correspond to any of the following
     mathematical entity types:

- Definition
- Theorem (this includes propositions, lemmas, corollaries)
- Example
- Exercise
- Proof

Only extract genuine mathematical content of these types.
Ignore and do NOT extract:

- General exposition or narrative explanation
- Chapter titles, section headings, or indexes
- Transitional text or motivation
- References to earlier or later results without stating them


## What qualifies as an entity

### 1. Definitions (very important)

A definition does NOT require a heading or numbering.

You must extract a block as a definition if its primary purpose is to introduce, name, or precisely describe a
      mathematical object or concept, even if it appears inline in the text.

Treat a block as a definition if it contains definitional language, including but not limited to:

- "is called"
- "is defined as"
- "we call"
- "is said to be"
- "the ... of ... is"
- "an element ... is"
- an emphasized or italicized term followed by an explanation
- a mathematical object followed by a copular verb ("is", "are") and a precise condition

If a paragraph answers the question "What is X?", it is a definition.

A single paragraph may introduce multiple concepts (e.g., minimum polynomial and conjugate).
If so, extract the entire paragraph as one definition entity.

Unlabeled definitions are mandatory.
Failure to extract them is considered an incorrect output.


### 2. Theorems

Extract any explicitly stated mathematical claim intended to be true, including:

- Theorem
- Proposition
- Lemma
- Corollary

Treat all of these as type "theorem".

The entity must assert a result, not merely mention one.
```

```
### 3. Proofs

Extract a proof only when:

- It is explicitly labeled (e.g., "Proof.", "Proof of Proposition 3.2."), and
- It contains actual mathematical reasoning or argument

Special rule:
If a proof appears only as part of an exercise prompt (e.g., "Prove that ...") and contains no mathematical
     argument beyond the statement itself, do NOT extract a separate proof entity.

Caution:
Be careful to extract a full proof, and do not cut off any information halfway through.


### 4. Examples

Extract examples that demonstrate or instantiate a mathematical concept, whether labeled ("Example 2.3") or
     clearly signposted in the text.


### 5. Exercises

Extract exercises or problems intended for the reader to solve, whether labeled or clearly formatted as such.


## Output format

For each extracted entity, produce a JSON object with the following keys:

- "type": one of "definition", "theorem", "example", "exercise", "proof" (always lowercase)
- "identifier": a concise label taken verbatim from the text introducing the entity
  (e.g., "Definition 6.9.", "Proposition 6.11.", "Proof.", "Exercise 7(b)")
  If no explicit label, number, or name is present, set this to null.
- "text": the exact text of the entity, copied verbatim.
  Do NOT summarize, paraphrase, normalize spacing, or rewrite mathematics.


## Ordering rules

- Preserve the original order of appearance.
- Each entity must be its own JSON object.
- Do not merge separate entities.
- Do not invent or infer missing text.


## Reasoning step (required)

Before producing the final output:

1. Scan the text systematically from start to end.
2. Actively search for:
   - Explicit labels (Definition, Theorem, Proof, etc.)
   - Implicit definitions using definitional language
3. For each candidate block:
   - Decide whether it is genuine mathematical content
   - Confirm it matches one of the allowed entity types
4. Exclude all non-mathematical exposition.

Only after completing this identification and segmentation should you produce the output.


## Output constraints

- Respond with a JSON list only
- Do NOT wrap the JSON in code fences
- Do NOT include explanations, commentary, or reasoning in the output
- If no valid entities are found, return an empty list: []


## Priority reminder

If a block introduces or defines a mathematical object - even without a heading - it MUST be extracted as a "
     definition".
```

Figure 6: Prompt for entity extraction

```
You are given a mathematical definition written in textbook style.

Your task is to extract the names of the mathematical objects, concepts, or properties being defined.

### Rules:
- Return only the names of the items being defined.
- Do not include symbols, formulas, or explanations.
- If multiple items are defined, return all of them.
- If a definition gives multiple synonymous names (e.g. "X (or Y)"), include the most standard or commonly
     used name.
- Ignore purely notational conventions unless they define a commonly named concept.
```

```
- Normalize names to lowercase unless capitalization is standard.
- Output the result as a JSON-style list of strings.

Definition:
{{DEFINITION_TEXT}}

Output format:
["name 1", "name 2", ...]

### Examples:

Example 1:
Definition:
**Definition 0.3.4**.:
1. A _union_ of two sets \(A\) and \(B\) is defined as \[A\cup B\coloneqq\{{x:x\in A\text{{ or }}x\in B\}}.\]
2. An _intersection_ of two sets \(A\) and \(B\) is defined as \[A\cap B\coloneqq\{{x:x\in A\text{{ and }}x\in
      B\}}.\]
3. A _complement of \(B\) relative to \(A\)_ (or _set-theoretic difference_ of \(A\) and \(B\)) is defined as
      \[A\setminus B\coloneqq\{{x:x\in A\text{{ and }}x\notin B\}}.\]
4. We say _complement_ of \(B\) and write \(B^{{c}}\) instead of \(A\setminus B\) if the set \(A\) is either
     the entire universe or if it is the obvious set containing \(B\), and is understood from context.
5. We say sets \(A\) and \(B\) are _disjoint_ if \(A\cap B=\emptyset\)

Output:
["union", "intersection", "set-theoretic difference", "complement", "disjoint"]

Example 2:
Definition:
**Definition 6.1** (\(\mathsf{{RCA}}_{{0}}\)).: An _integral domain_ is a ring \(R\) with no nonzero zero-
     divisors

Output:
["integral domain"]
```

Figure 7: Prompt for name extraction

```
You are an expert mathematician and a specialist in formal mathematics, especially Lean 4 and mathlib4.

Your task is to identify all mathematical references that appear in a given mathematical definition, theorem,
     lemma, or statement.

A reference is any term, symbol, or phrase whose meaning depends on a previously defined mathematical object,
     entity, or external context.

You must classify each reference into exactly one of the following categories.

--------------------------------------------------
REFERENCE TYPES
--------------------------------------------------

1. "object_reference"

A standard, named mathematical object, structure, property, or concept whose definition exists independently
     of the current text.

Examples:
- group
- topological space
- metric space
- compact space
- real number
- real vector space
- group homomorphism
- general linear group
- continuous function
- prime number
- finite set

Include:
- Named mathematical structures
- Named mathematical properties if they have formal definitions
- Standard named constructions

Do NOT include:
- Generic words like "element", "set", "function", "number"
- Logical words like "if", "then", "and", "exists"
- Less specific forms of an object. I.e., in the sentence "Let G be an abelian group", you should include "
     abelian group", NOT "group".

--------------------------------------------------

2. "entity_reference"

A reference to a specific previously defined, labeled, or numbered mathematical item.

Examples:
- Definition 3.4
- Theorem 2.1
- Lemma 5
- Proposition 4.2
```

```
- Corollary 1.3
- previous lemma
- above theorem
- following definition

These refer to specific formal entities in the document structure.

--------------------------------------------------

3. "local_variable_reference"

A symbol that refers to a mathematical object but is NOT explicitly introduced in the current text.

A symbol is explicitly introduced if the current text declares it using phrases such as:

- let M be ...
- suppose M is ...
- fix M ...
- given M ...
- where M is ...
- for M a ...
- denote by M ...
- let M := ...
- assume M is ...
- take M ...
- any equivalent explicit declaration

If a symbol is explicitly introduced in the current text, DO NOT include it.

Only include symbols that are used WITHOUT being introduced.

EXAMPLES

Example:

Text:
"Let G be a group. Then G is abelian iff the center of G equals G."

Reasoning:
- G is explicitly introduced -> exclude
- group is a named object -> include
- abelian is a named property -> include
- center is a named construction -> include

Output:
[
  {"term": "group", "reference_type": "object_reference"},
  {"term": "abelian", "reference_type": "object_reference"},
  {"term": "center", "reference_type": "object_reference"}
]

--------------------------------------------------

Example:

Text:
"M is compact iff G is abelian."

Reasoning:
- M not introduced -> include local_variable_reference
- G not introduced -> include local_variable_reference
- compact is a named property -> include object_reference
- abelian is a named property -> include object_reference

Output:
[
  {"term": "M", "reference_type": "local_variable_reference"},
  {"term": "G", "reference_type": "local_variable_reference"},
  {"term": "compact", "reference_type": "object_reference"},
  {"term": "abelian", "reference_type": "object_reference"}
]

--------------------------------------------------

If no references exist, output:

[]
```

Figure 8: Prompt for reference extraction

```
You are a meticulous expert in Lean 4 and `mathlib4`.

Your task is to act as a "grounding" reasoner for a formalization agent. Your goal is to determine whether a
      given mathematical concept has a canonical formal definition in `mathlib`, based on a list of search
      candidates.

--------------------------------------------------
Your Task (Follow These Steps PRECISELY)
--------------------------------------------------
```

```
--------------------------------------------------
Step 1: Direct Match Analysis
--------------------------------------------------

- First, look for a **direct, canonical definition** among the candidates.
- A direct match is typically a `class`, `structure`, or `def` whose name is very similar to the concept name.

  Example:
  - Concept: "local ring"
  - Match: `class IsLocalRing`

- If you find a clear, direct match, use that as your primary answer.

--------------------------------------------------
Step 2: Deduction from Usage Patterns
(Only if no direct match is found in Step 1)
--------------------------------------------------

- If no direct match was found in Step 1, your task is to **deduce** the canonical name by finding a **
     consistent usage pattern** across multiple `theorem` and `instance` candidates.

- **Analyze the signatures:**
  Look for a common identifier that is consistently used as a **type** or **typeclass** across multiple
       candidates.

- **Example:**
  If you are looking for "CharZero" and the search results include:
  - `instance : CharZero N`
  - `instance : CharZero Z`
  - `theorem my_thm [CharZero R]`

  Then the identifier `CharZero` appears repeatedly as a typeclass. This is overwhelming evidence that the
       canonical definition is named `CharZero`.

- **Strict Rule:**
  The name you select **must** be an identifier that is explicitly present in the candidate list.

  - Do **not** invent, combine, or guess a new name.
  - If no single, consistent pattern emerges from the candidates, you must conclude that no confident match
       can be found.

--------------------------------------------------
Step 3: Final Decision
--------------------------------------------------

- Based on your analysis from Step 1 and Step 2, determine the single best candidate for the concept.

- Your answer MUST be a single, valid JSON object with the following keys:

  - `"best_match"`:
    The **list index** of the candidate which precisely matches the search query. If no confident match can be
         found through either direct matching or inference, the value must be `null`.

  - `"reasoning"`:
    An explanation of why you chose a match, or didn't choose any match.
```

Figure 9: Prompt for grounding entities to mathlib. 10 candidate results are fed into this prompt, and the output is an index of the best match and a reasoning.

C Additional Experimental Details

We use vLLM (Kwon et al., 2023) as an efficient inference server for all of our dataset construction, autoformalization, and evaluation. If specified, we use the author's recommended sampling parameters. Otherwise, we use a temperature of 0.5, a top-p of 0.95, and otherwise default parameters for all autoformalization experiments. We use deterministic sampling for our semantic faithfulness systems.

All experiments were run on a single H200 GPU. Depending on the model, full-benchmark runs run from 1-10 hours. Checking compilation runs on 10 cpu workers for approximately 5 minutes. Measuring faithfulness runs for $\sim$1-2 hours, depending on the number of successfully compiling examples.

D Broader impacts

We release *MathAtlas* with the hope to advance the field of autoformalization, and with it, AI and mathematics in general. This outcome can potentially have larger societal impact as AI becomes more powerful and more widely used. However, it is unlikely that this contribution in particular will have large-scale negative ramifications.

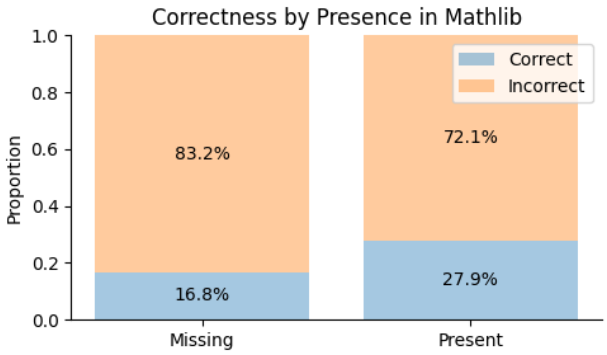


Figure 10: Correctness of formalized entities from our best models compared with presence in Mathlib. Entities which our retrieval system (LeanSearch, (Gao et al., 2025a)) found in Mathlib are significantly more likely to be correctly formalized than otherwise.

Table 5: Token costs for Claude Sonnet and Claude Code experiments on *MA-Hard*.

Setting	Calls	Tokens/Item	$/item	Total $	Correctness
Sonnet	1.0	14.0k	0.031	21.76	27.6%
+ deps in context	1.0	58.3k	0.121	84.18	29.5%
+ iterative	1.8	36.1k	0.079	55.02	50.7%
Claude Code	5.4	164.6k	0.146	101.61	57.4%
+ dependency aware	6.9	235.0k	0.191	133.11	64.1%

E LIMITATIONS

MathAtlas primarily deals with graduate-level mathematics, and leaves more difficult domains such as research mathematics for future work. Certain textbooks are also omitted from *MathAtlas* due to copyright concerns, which can potentially bias data. We perform an intrinsic data quality analysis of *MathAtlas* and find high-quality extractions. However, some noise still exists. Finally, compute and price concerns limit our study of the dataset.

Table 6: Autoformalization results on the "open" split of MathAtlas. The results listed here are similar to those in Table 3, but restricted to a subset of the data.

Model	Compiles	Faithful	Correct
Prompted			
gpt-oss-20b (zs)	9.5%	48.4%	4.6%
+few-shot	16.3%	47.2%	7.7%
+tuned prompt	16.8%	47.6%	8.0%
+tuned exs.	19.9%	60.3%	12.0%
gpt-oss-120b (zs)	13.0%	49.2%	6.4%
+few-shot	24.2%	57.0%	13.8%
+tuned prompt	26.0%	58.8%	15.3%
+tuned exs.	**27.4%**	**62.0%**	**17.0%**

(a) Autoformalization results for definitions. The best performance in each category and column are bolded.

Model	Compiles	Faithful	Correct
Fine-tuned			
Herald 7B	11.8%	14.4%	1.7%
Kimina 7B	**27.1%**	8.1%	2.2%
ATLAS-L 8B	20.1%	16.9%	3.4%
Goedel 8B	18.9%	37.6%	7.1%
Goedel 32B	22.3%	32.7%	7.3%
ReForm 8B	18.1%	**50.8%**	**9.8%**
Prompted			
gpt-oss-20b (zs)	9.3%	49.5%	4.6%
+few-shot	16.5%	35.8%	5.9%
+tuned prompt	16.8%	32.7%	5.5%
+tuned exs.	19.1%	35.1%	6.7%
gpt-oss-120b (zs)	15.5%	43.2%	6.7%
+few-shot	16.3%	**44.2%**	7.2%
+tuned prompt	17.1%	43.9%	7.5%
+tuned exs.	**20.7%**	38.6%	**8.0%**

(b) Autoformalization results for statements (theorems, examples, and exercises). The best performance in each category and column are bolded.