Datasets, Formalizers, and Provers: A Survey of Formal Mathematical Reasoning with LLMs

Anonymous ACL submission

Abstract

Recent advances in large language models (LLMs) have significantly expanded the capabilities of automated mathematical reasoning. This survey reviews recent progress in formal mathematical reasoning with LLMs from three interconnected perspectives: datasets, formalizers, and provers. We provide a comparative analysis of key benchmarks from 2022 to 2026, highlighting dataset properties and model performance trends. We argue that future progress will require richer and more diverse benchmarks, evaluation protocols that emphasize semantic correctness and robustness, and deeper alignment between informal mathematical intuition and formal symbolic structure.

1 Introduction

Large language models (LLMs) have substantially advanced automated mathematical reasoning, from solving natural-language word problems to interacting with formal proof assistants. Early progress largely centered on informal problem solving, where models generate free-form solutions to problems stated in natural language or LaTeX. Benchmarks such as GSM8K (Cobbe et al., 2021) and MATH (Hendrycks et al., 2021), together with prompting methods such as chain-of-thought reasoning (Wei et al., 2022), zero-shot CoT (Kojima et al., 2022), and analogical reasoning (Zhang et al., 2023), demonstrated nontrivial mathematical reasoning behavior in LLMs. However, informal solutions remain difficult to verify reliably: a seemingly fluent derivation may still contain hidden logical gaps, invalid steps, or hallucinated arguments.

This limitation has motivated growing interest in *formal mathematical reasoning*, where statements and proofs are expressed in formal languages and checked by proof assistants such as Lean (de Moura et al., 2015), Isabelle/HOL (Nipkow et al., 2002), Coq (The Coq Development Team, 1999–), and Metamath (Megill and Wheeler, 2019). In this setting, correctness is enforced by machine checking, and feedback from the proof environment can guide generation, repair errors, and train stronger systems. LLMs play two central roles in this pipeline. **Formalizers** translate informal mathematical statements into machine-checkable specifications, with recent examples including Kimina-Autoformalizer (Wang et al., 2025a) and RAutoFormalizer (Liu et al., 2025b). **Provers** generate or search for formal proofs, often using proof-assistant feedback, retrieval, reinforcement learning, or agentic orchestration; representative systems include DeepSeek-Prover-V2 (Ren et al., 2025), Kimina-Prover (Wang et al., 2025a), Gödel-Prover-V2 (Lin et al., 2025), AlphaProof (Hubert et al., 2025), and Seed-Prover (Chen et al., 2025c,b).

The rapid growth of this area has produced a diverse ecosystem of datasets, formalizers, and prover systems. Benchmarks now range from informal arithmetic and Olympiad problems to Lean-verified college mathematics, graduate-level autoformalization, applied mathematics, and real-world proof-engineering tasks. At the same time, progress remains difficult to interpret: dataset coverage is uneven, formalization can preserve syntax while losing meaning, and prover success often depends on search budget, library snapshots, retrieval quality, and proof-assistant versions. These issues make formal mathematics a rigorous but demanding testbed for studying verifiable reasoning with LLMs.

In this survey, we focus on formal mathematical reasoning with LLMs and organize recent work along three axes:

1. **Datasets**: benchmarks and corpora for training and evaluating formalization and proof synthesis, with emphasis on mathematical diversity, abstraction depth, and library dependence.

2. **Formalization Methods**: models and pipelines for translating informal mathematics into machine-checkable formal statements while preserving mathematical intent.
3. **Provers and Integration Strategies**: LLM-driven proof generation methods, including tactic synthesis, guided search, retrieval-augmented reasoning, proof repair, reinforcement learning, and agentic proving.

Our contributions are fourfold:

- We introduce a structured taxonomy of LLM-based formal mathematical reasoning across datasets, formalizers, prover models, and prover pipelines.
- We provide an updated survey of major benchmarks, formalization systems, and theorem provers through 2026, highlighting dataset properties, methodological trends, and evaluation practices.
- We synthesize recurring challenges across the formal reasoning pipeline, including dataset depth, semantic faithfulness, library grounding, search-budget comparability, and reproducibility.
- We distill practical recommendations for future datasets, formalizers, and provers, including depth-aware benchmark design, semantic-faithfulness checks, dependency-aware evaluation, and budget-normalized prover reporting.

2 Datasets Perspective

The evolution of LLM-based mathematical reasoning can be traced through datasets that progressively shift from informal arithmetic problem solving to fully formal theorem proving. Along this trajectory, dataset design has co-evolved with domain complexity, degrees of formalization, and evaluation paradigms, reflecting increasingly ambitious goals for machine reasoning.

Natural-language mathematical reasoning. Early benchmarks elicited step-by-step reasoning in natural language. **GSM8K** (Cobbe et al., 2021) contains approximately 8,500 grade-school word problems requiring short multi-step arithmetic reasoning, and played a central role in motivating chain-of-thought prompting. **MATH** (Hendrycks et al., 2021) raises the difficulty with 12,500 competition-style problems across algebra, geometry, precalculus, number theory, and counting/probability, each annotated with solutions and difficulty levels. **OmniMATH** (Gao et al., 2024) extends this line to over 4,500 Olympiad-level problems from multilingual and multi-source settings, challenging models on higher-level mathematical reasoning and cross-cultural generalization. Although these benchmarks are primarily informal, they have also served as sources for formalization and proof-generation studies.

Transition to formal theorem proving. **MiniF2F** (Zheng et al., 2021) marks a decisive transition toward formal reasoning, with 488 high-school and competition problems formalized across Lean, Metamath, Isabelle, and HOL Light. By shifting evaluation from textual explanation to machine-checkable proof synthesis, MiniF2F laid the foundation for neural theorem-proving research. **ProofNet** (Azerbayev et al., 2023) expands this direction with 3,675 Lean theorems paired with human-readable proofs from undergraduate mathematics, exposing large performance gaps in neural provers and establishing *ProofNet-Hard* as a stress test.

Several datasets push formal reasoning into harder competition mathematics. **FIMO** (Liu et al., 2023) formalizes IMO and national Olympiad problems in Lean 4 across algebra, geometry, number theory, and combinatorics, with emphasis on invariants and extremal arguments. **PutnamBench** (Tsoukalas et al., 2024) formalizes Putnam competition problems spanning analysis, algebra, number theory, combinatorics, and topology, highlighting the gap between informal competition statements and fully verified proofs.

Other resources broaden the style and scale of formal mathematical data. **IsarStep** (Li et al., 2021) provides Isabelle/Isar-style structured proofs, emphasizing human-like proof steps rather than low-level tactic scripts. **LeanWorkbook** (Ying et al., 2024) scales curricular coverage to over 20,000 Lean exercises from undergraduate textbooks, spanning linear algebra, analysis, algebra, topology, geometry, probability, graph theory, number theory, and combinatorics. **RFMDataset** (Guo et al., 2025a) takes a diagnostic view, curating problems to catalogue LLM failure modes such as hallucinations, logical inconsistencies, and invalid proof steps.

Large-scale and autoformalization-oriented

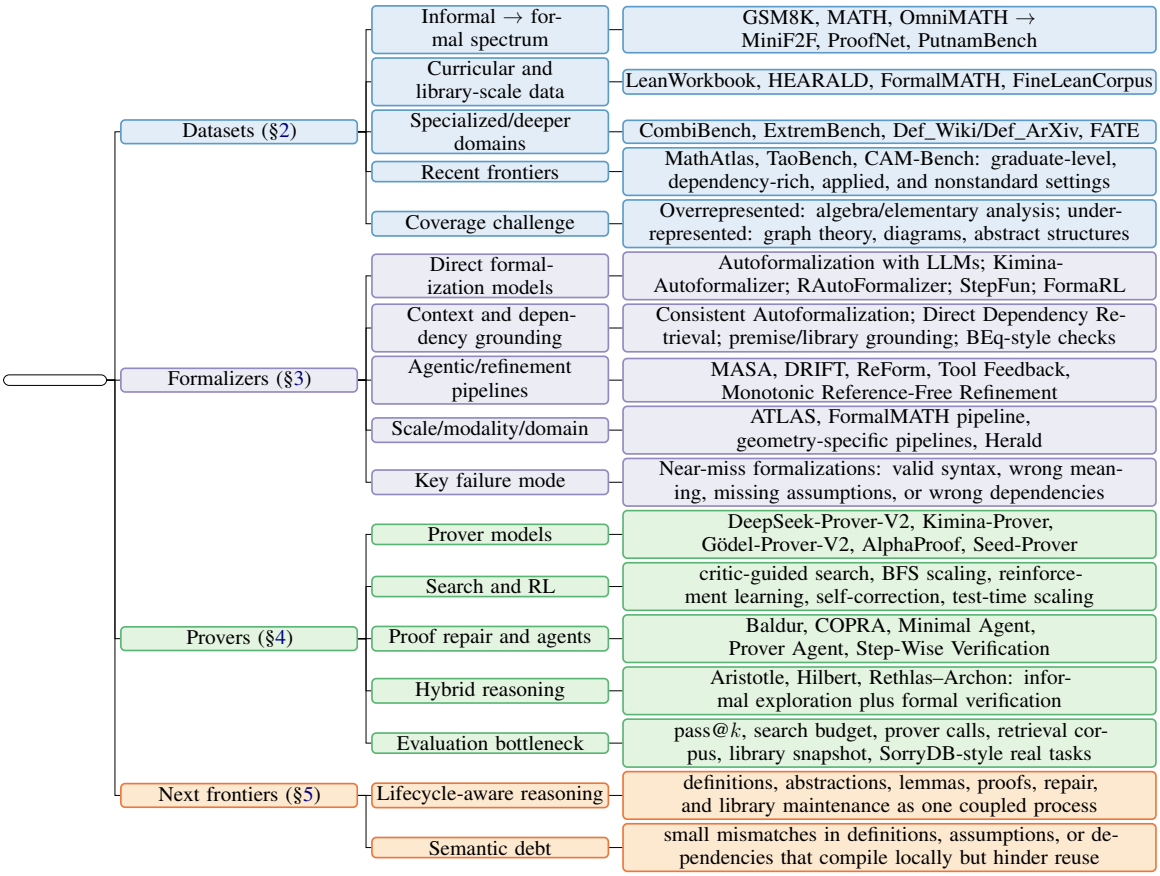

Figure 1: Taxonomy of LLM-based formal mathematical reasoning. Datasets, formalizers, and provers form an interconnected pipeline: datasets define the mathematical and library context, formalizers translate informal intent into formal statements, and provers search for machine-checkable proofs. Evaluation protocols and lifecycle-aware reasoning determine whether progress is robust, reproducible, and useful beyond static benchmark completion.

datasets further extend the landscape. **HEARALD** (Gao et al., 2025) pairs natural-language annotations with Lean 4 tactics across Mathlib, supporting tactic alignment and human-style commentary. **FormalMATH** (Yu et al., 2025) contains 5,560 Lean 4 verified college-level statements constructed through a human-in-the-loop autoformalization pipeline, showing that strong models still achieve limited success under practical sampling budgets. **FineLeanCorpus** (Peng et al., 2025) provides nearly 286,000 verified natural-language statements paired with Lean 4 formalizations, making it a major resource for training and evaluating autoformalizers.

Recent specialized benchmarks expose finer-grained reasoning gaps. **CombiBench** (Liu et al., 2025a) focuses on combinatorial reasoning with 100 Lean 4 problems and a fine-grained evaluation framework beyond binary success. **ExtremBench** (Gao and Han, 2025) targets 93 extremal optimization problems derived from Chinese Mathematical Olympiad inequality exercises, evaluating maxima/minima reasoning under constraints. **Def_Wiki** and **Def_ArXiv** (Zhang et al., 2025a) shift attention from proofs to definitions, emphasizing ambiguous, context-dependent formalization and library grounding. **FATE** (Jiang et al., 2025) targets abstract and commutative algebra from undergraduate exercises to PhD-level qualifying problems, testing advanced algebraic reasoning beyond standard contest mathematics.

The newest benchmarks increasingly move beyond Olympiad-style and undergraduate settings. **MathAtlas** (Patel et al., 2026) targets in-the-wild graduate-level autoformalization, collecting roughly 52K mathematical objects from 103 graduate textbooks with rich contextual and dependency annotations. **TaoBench** (Taylor et al., 2026) evaluates whether Lean-based ATP systems generalize beyond standard Mathlib definitions using exercises from Terence Tao's formalization of *Analysis I*, where core concepts are developed from first principles. **CAM-Bench** (Long et al., 2026) extends formal theorem-proving evaluation to compu-

tational and applied mathematics, providing 1,000 Lean 4 proof targets across optimization, numerical linear algebra, and numerical analysis.

Overall, existing datasets reveal three trends. First, the field has shifted from answer-based natural-language reasoning toward machine-checkable formalization and proof synthesis. Second, many benchmarks remain concentrated around competition mathematics, algebra, and elementary analysis, leaving modern graph theory, applied mathematics, diagrammatic geometry, and highly context-dependent research mathematics underrepresented. Third, recent datasets increasingly recognize that formal reasoning requires more than a problem statement: reliable evaluation must expose dependencies, definitions, local context, proof-assistant versions, and semantic-faithfulness criteria. These trends motivate future benchmarks that are not merely larger, but deeper, more domain-balanced, and more explicit about the mathematical and library context required for verification.

3 Formalizers Perspective

Autoformalization systems, or "formalizers," bridge informal mathematical language and machine-verifiable languages such as Lean, Isabelle/HOL, and Coq. We organize recent work into two families: **specialized formalization models**, trained for direct informal-to-formal translation, and **formalization agents or pipelines**, which combine LLMs with retrieval, compiler feedback, semantic checking, and iterative refinement. Across both families, failures typically arise from syntactic invalidity, semantic drift, missing or hallucinated dependencies, and sensitivity to informal phrasing; thus, the central challenge is preserving mathematical intent rather than merely producing code that compiles.

3.1 Formalization Models

Early work on **Autoformalization with Large Language Models** (Wu et al., 2022) established autoformalization as a distinct LLM task, showing that generated formal statements can improve downstream theorem proving and motivating later Lean- and Isabelle-based systems using supervised learning, retrieval, compiler feedback, and semantic validation.

Recent specialized models improve direct translation through stronger data curation, evaluation, and verifiable feedback. **Kimina-Autoformalizer** (Wang et al., 2025a) uses curated informal–formal pairs, expert iteration, Lean 4 compilation filtering, LLM-based semantic judging, contradiction detection, and negation testing, showing that compilation alone is insufficient for semantic faithfulness. **RAutoFormalizer** (Liu et al., 2025b) addresses unreliable evaluation and limited library awareness through **BEq** (Bidirectional Extended Definitional Equivalence) and dependency-aware retrieval, showing that autoformalization requires both semantic alignment and grounding in existing definitions.

Other models emphasize reasoning, reinforcement learning, and data efficiency. **StepFun-Formalizer** (Wu et al., 2025) combines supervised fine-tuning on large-scale informal–formal pairs with reinforcement learning from verifiable rewards and reasoning-augmented traces; its weaker performance on CombiBench indicates that combinatorial and structurally complex domains remain difficult. **FormaRL** (Huang et al., 2025) studies reinforcement learning on unlabeled problems, deriving rewards from Lean compiler feedback and LLM-based semantic consistency, suggesting that verifiable feedback can partly compensate for scarce informal–formal pairs.

Overall, specialized models have advanced through fine-tuning, filtering, and reinforcement learning, but remain most reliable on curated benchmark distributions and less robust when inputs require implicit context recovery, nonstandard definitions, or conventions that differ from training data.

3.2 Formalization Agents and Pipelines

Agentic formalization pipelines decompose autoformalization into retrieval, syntactic checking, semantic critique, and iterative refinement. This design reflects a key limitation of single-pass translation: many failures arise from missing context, incorrect definitions, or poor library grounding rather than surface syntax alone.

Several systems make context construction and library coherence central to the formalization process. **MASA** (Zhang et al., 2025b) uses separate agents for formalization, syntactic critique, semantic critique, and refinement, while **DRIFT** (Zhang et al., 2025c) decomposes autoformalization into statement analysis, premise retrieval, example retrieval, and formal generation. **Consistent Autoformalization** (Zhang et al., 2024) shifts attention

from isolated statement translation to library-scale coherence, requiring formalizations to remain syntactically valid, terminologically consistent, and semantically aligned across related mathematical objects. **Direct Dependency Retrieval** (Wang et al., 2025c) further improves grounding by predicting candidate formal dependencies from the informal statement and verifying their existence in the target library, reducing hallucinated identifiers.

Other pipelines focus on refinement under tool feedback. **ReForm** (Chen et al., 2025a) uses prover-derived value estimates to reduce irreversible generation errors. **Autoformalizer with Tool Feedback** (Guo et al., 2025b) combines live Lean feedback, LLM-based semantic judging, synthetic pretraining, and preference-based optimization. **Monotonic Reference-Free Refinement** (Zhang et al., 2026) extends this direction by refining candidate formalizations without a gold reference statement, combining LLM-based assessment with prover feedback for full-theorem autoformalization.

Finally, several systems address scale, modality, or domain specificity. **ATLAS** (Liu et al., 2025c) constructs large-scale synthetic informal–formal data from formal libraries with teacher–student distillation and expert revision. **FormalMATH** (Yu et al., 2025) uses a human-in-the-loop pipeline to construct Lean 4 verified statements from college-level mathematics. Geometry-specific pipelines (Murphy et al., 2024) combine diagrammatic reasoning, symbolic geometry engines, and SMT-based equivalence checking to recover implicit geometric constraints, while the **Herald** pipeline (Gao et al., 2025) constructs natural-language annotated Lean data from Mathlib using retrieval-augmented, section-level translation.

Recent diagnostic work shows that robustness cannot be measured only by compilation or a single canonical phrasing. **Surface Sensitivity in Lean 4 Autoformalization** (Feng et al., 2026) finds that semantically equivalent paraphrases can produce divergent formal outputs, with many failures occurring at the compilation boundary. Future evaluations should therefore report paraphrase robustness, compile-conditional consistency, and semantic equivalence, rather than relying on compilation alone.

4 Provers Perspective

LLM-based theorem provers generate or search for machine-checkable proofs in proof assistants such as Lean, Isabelle/HOL, and Coq. Compared with formalizers, provers face long-horizon planning, sparse feedback, large tactic spaces, dependence on retrieved lemmas, and sensitivity to proof-assistant versions and library snapshots. Recent work has therefore moved from single-pass tactic generation toward feedback-driven search, proof repair, reinforcement learning, retrieval augmentation, and agentic proof orchestration.

4.1 Prover Models

Early LLM-based provers focused on generating short tactic sequences or high-level proof sketches. **Draft–Sketch–and–Prove** (Jiang et al., 2022) showed that informal proof sketches can guide formal proof construction, but also exposed the difficulty of scaling sketch-based methods to longer, library-dependent proofs.

Subsequent systems increasingly treat proof search as a learned decision process. **InternLM2.5-StepProver** (Wu et al., 2024) trains a critic model from prover feedback to score partial proof states and guide best-first search. **BFS-Prover** (Xin et al., 2025) combines breadth-first exploration, GPU-accelerated Lean compilation, and learned pruning heuristics. Together, these systems reflect a shift from direct generation toward search policies guided by proof-state feedback.

More recent provers emphasize synthetic data generation, verifier feedback, and reinforcement learning tightly coupled with the proof assistant. **DeepSeek-Prover-V2** (Ren et al., 2025) trains on large-scale self-generated Lean 4 proof trajectories and refines its policy using compiler feedback. **Kimina-Prover** (Wang et al., 2025a) focuses on competition-style mathematics and human-readable Lean proofs, making it useful both as a standalone prover and as a base model for interactive pipelines. **Gödel-Prover-V2** (Lin et al., 2025) further illustrates this trend through open-source Lean models that combine whole-proof generation, verifier feedback, and iterative self-correction.

A particularly visible reinforcement-learning approach is **AlphaProof** (Hubert et al., 2025), which combines formal proof search with reinforcement learning over a proof environment. Its success on IMO-style problems demonstrates the potential of RL for formal mathematical reasoning, while

also highlighting the substantial computational cost and test-time scaling often required by frontier systems. Similarly, **Seed-Prover** and **Seed-Prover 1.5** (Chen et al., 2025c,b) combine large-capacity models with Lean-guided self-critique, lemma invention, reinforcement learning, and efficient test-time scaling. However, limited disclosure in some frontier systems makes reproducibility and fair comparison difficult.

4.2 Prover Pipelines

Beyond standalone models, many recent systems organize proving as a pipeline involving retrieval, planning, critique, verification, repair, and backtracking. This is motivated by a central limitation of direct generation: a proof attempt may fail not because the model lacks a tactic, but because it lacks the right lemma, abstraction, or decomposition strategy.

Baldur (First et al., 2023) introduced whole-proof generation and repair for Isabelle/HOL, using a repair model conditioned on failed proof attempts and prover error messages. This makes Baldur an important precursor to later feedback-driven proof repair systems. **COPRA** (Thakur et al., 2023) represents an agentic in-context learning approach: a general-purpose LLM proposes tactics within a stateful backtracking loop, while the proof assistant executes each tactic and returns feedback. COPRA shows that proof search can be orchestrated through prompting, environment feedback, and search history without fully fine-tuning a prover for each environment. Recent work on **A Minimal Agent for Automated Theorem Proving** (Requena Pozo et al., 2026) further argues for simple, reproducible agentic baselines that expose the contribution of iterative refinement, library search, and context management.

Classical automated theorem provers have also been integrated with LLM guidance. **ProofCompass** (Wischermann et al., 2025) combines LLM-generated high-level proof sketches with ATP systems such as Vampire or E, using learned premise selection to connect neural suggestions with symbolic automation. Other pipelines emphasize verification-in-the-loop and role decomposition. **Prover Agent** (Baba et al., 2025) decomposes proving into interacting roles such as planner, tactician, verifier, and critic, while **Step-Wise Formal Verification** (Zhou and Zhang, 2025) verifies each generated step to prevent error accumulation in long proofs.

Retrieval-augmented pipelines address the growing importance of library grounding. **REAL-Prover** (Shen et al., 2025) and related systems integrate dense premise retrieval and reranking to supply relevant lemmas during proof search. **Aria** (Wang et al., 2025b) extends this direction by maintaining dependency graphs across larger proof projects, supporting coherent translation and verification of multi-page mathematical texts.

Recent hybrid systems combine formal search with informal mathematical reasoning. **Aristotle** (Achim et al., 2025) integrates Lean proof search, informal lemma generation and formalization, and a dedicated geometry solver for IMO-level theorem proving. **Automated Conjecture Resolution with Formal Verification** (Ju et al., 2026) extends this hybrid direction to research-level mathematics through two agents: Rethlas, which explores informal solution strategies using theorem search, and Archon, which translates and verifies candidate arguments as Lean 4 projects. This line of work suggests a broader shift from benchmark proof reconstruction toward systems that combine informal mathematical exploration with formal verification. Related systems such as **Hilbert** (Varambally et al., 2025) similarly attempt to translate human-style informal proofs into Lean by decomposing mathematical intuition into verifiable steps.

Evaluation caveat. Prover results are difficult to compare directly across papers because reported success can depend heavily on pass@k, search budget, timeout, number of prover calls, retrieval corpus, proof-assistant version, Mathlib snapshot, and whether the task begins from an already formalized statement or includes autoformalization. Recent benchmarks such as **SorryDB** (Letson et al., 2026), which collects open Lean tasks from real formalization projects, further show that practical prover usefulness is not fully captured by static competition-style benchmarks. Future evaluations should therefore report success rates together with search budget, library version, proof length, failure categories, and the boundary between formalization and proving.

5 Challenges and Discussion

Despite rapid progress, formal mathematical reasoning with large language models (LLMs) faces persistent challenges across the full pipeline, encompassing datasets, formalization, and automated

proving. Many limitations stem not from individual components, but from their interaction: gaps in dataset coverage lead to fragile formalizations, which in turn impose excessive demands on provers. We analyze these issues from three complementary perspectives and distill their practical implications.

Summary of takeaways. First, datasets should report not only mathematical domains but also abstraction level, prerequisite structure, and library dependencies. Second, formalizers should be evaluated beyond compilation, since syntactically valid statements may still suffer from semantic drift. Third, prover comparisons should normalize search budget, timeout, prover calls, proof-assistant version, and library snapshot. Finally, end-to-end systems should make clear whether a failure arises from formalization, retrieval, proof search, or genuine mathematical difficulty.

5.1 Dataset Coverage and Mathematical Breadth

Although recent benchmarks have grown significantly in size and ambition, they still cover only a narrow portion of mathematics. Most widely used datasets emphasize algebraic manipulation, elementary number theory, competition mathematics, and classical analysis. Benchmarks such as GSM8K (Cobbe et al., 2021), MATH (Hendrycks et al., 2021), LeanWorkbook (Ying et al., 2024), and FormalMATH (Yu et al., 2025) illustrate this trend: even when multiple domains are nominally included, many problems remain close to familiar algebraic, analytic, or textbook-style patterns.

The limitations of this coverage become particularly clear when considering graph theory and advanced combinatorics. Several datasets include combinatorics or graph-related labels, yet this coverage is often shallow relative to the breadth of modern discrete mathematics. Problems involving flows, matchings, graph minors, spectral methods, probabilistic methods, or structural decomposition—central topics in modern graph theory—remain largely absent from commonly used formal-reasoning benchmarks. Similarly, while CombiBench (Liu et al., 2025a) provides valuable fine-grained evaluation for combinatorial reasoning, it cannot by itself represent the full abstraction level of research-level graph theory and combinatorics.

Recent benchmarks begin to address this gap. MathAtlas (Patel et al., 2026) moves toward graduate-level, dependency-rich mathematical objects; TaoBench (Taylor et al., 2026) tests reasoning under nonstandard foundational definitions; CAMBench (Long et al., 2026) introduces computational and applied mathematics; and FATE (Jiang et al., 2025) targets advanced algebra. These efforts suggest that future progress should not be measured only by benchmark size. A smaller dataset with deeper abstraction, explicit dependencies, and underrepresented domains may be more informative than a larger collection of template-like exercises.

As a result, models trained on current benchmarks risk overfitting to a narrow style of mathematics. Strong performance may indicate mastery of benchmark-specific patterns rather than genuine mathematical generality. Expanding dataset coverage, both in terms of domains and in the depth at which they are represented, is therefore essential for assessing whether formal reasoning systems learn transferable mathematical principles.

5.2 Why Formalization Remains Hard

Formalizing mathematics is challenging even for human experts, and this difficulty is clearly reflected in current autoformalization systems. Informal mathematical writing often depends on implicit assumptions, shared conventions, overloaded notation, and unstated context, all of which must be recovered and made explicit in a rigid formal language. Bridging this gap remains one of the hardest steps in the pipeline.

Most modern systems adopt a strategy based on iterative feedback. A model proposes a candidate formalization, receives syntactic feedback from the proof assistant and semantic feedback from equivalence checks or LLM judges, and then revises its output. This pattern underlies many recent approaches, including Kimina-Autoformalizer (Wang et al., 2025a), RAutoFormalizer (Liu et al., 2025b), DRIFT (Zhang et al., 2025c), MASA (Zhang et al., 2025b), and tool-feedback-based pipelines (Guo et al., 2025b).

Although effective on curated benchmarks, this approach has important limitations. Iterative feedback can promote local repairs rather than faithful reconstruction of mathematical intent. A statement may compile while changing quantifier scope, omitting assumptions, selecting an unintended definition, or proving a weaker theorem. Such near-miss formalizations are especially problematic because errors may surface only during later proof attempts, making them hard to diagnose.

Recent work on paraphrase robustness reinforces this concern. Surface-sensitive autoformalization shows that semantically equivalent informal statements can lead to divergent formal outputs (Feng et al., 2026). Thus, future formalizer evaluation should distinguish compilation success, semantic equivalence, dependency correctness, and robustness under paraphrase or domain shift. Progress beyond syntactic repair loops will likely require richer semantic representations, stronger modeling of mathematical intent, and tighter integration between informal reasoning and formal libraries.

5.3 Proving Beyond Benchmarks

On the prover side, recent progress has been substantial. Systems such as DeepSeek-Prover-V2 (Ren et al., 2025), Kimina-Prover (Wang et al., 2025a), AlphaProof (Hubert et al., 2025), and Seed-Prover (Chen et al., 2025c,b) achieve strong results on benchmarks such as MiniF2F, ProofNet, and competition-style Lean datasets, showing that LLM-guided proof search can reconstruct many established proofs.

However, these advances are costly and difficult to compare directly. Proof search often requires large pass@k values, repeated prover interaction, long timeouts, and aggressive test-time scaling. Reported success can also depend on retrieval corpus, proof-assistant version, Mathlib snapshot, and whether the task begins from an already formalized statement or includes autoformalization. Benchmarks such as SorryDB (Letson et al., 2026), which collects open Lean tasks from real formalization projects, further suggest that static competition-style benchmarks may not fully capture practical prover usefulness.

This raises a key open question: *how well do current methods extend to mathematics that requires new abstractions rather than proof reconstruction?* Performance on research-level problems, where proofs depend on careful lemma selection, conceptual reformulation, or long-range structure, remains only partially tested. Closing this gap will require provers that operate across multiple levels of abstraction, maintain long-range coherence, and align search more closely with mathematical structure rather than surface patterns alone.

5.4 Next Frontiers

The next frontier is not simply larger benchmarks or stronger provers, but *lifecycle-aware formal reasoning*. Real mathematics evolves through a cycle of informal intuition, definition choice, lemma discovery, proof repair, and library maintenance. Current systems usually optimize isolated stages of this cycle, which creates *semantic debt*: small mismatches in definitions, assumptions, or dependencies that compile locally but later obstruct proof reuse and generalization. Future systems should therefore treat datasets, formalizers, and provers as coupled components of one evolving mathematical workspace.

Overall, progress should be measured not only by solved theorems, but by whether systems preserve intent, expose assumptions, manage dependencies, repair failures, and remain stable as formal libraries evolve.

6 Conclusion

This survey examined formal mathematical reasoning with large language models from the perspectives of datasets, formalizers, and provers. Recent progress has been substantial: benchmarks now extend beyond elementary and Olympiad-style mathematics, formalizers increasingly use retrieval and semantic feedback, and provers combine learned search, reinforcement learning, proof repair, and agentic orchestration. Together, these developments show that formal mathematics has become a central testbed for verifiable reasoning with LLMs.

At the same time, current systems remain specialized rather than general mathematical reasoners. Dataset coverage is still uneven, autoformalization can preserve syntax while losing mathematical intent, and prover performance is often sensitive to search budget, retrieval corpus, and proof-assistant versions. Future progress will therefore require deeper and more domain-balanced benchmarks, semantic-faithfulness checks, dependency-aware formalization, and reproducible prover evaluation under explicit resource budgets. More broadly, advancing formal mathematical reasoning will require tighter alignment between informal mathematical intuition and formal symbolic verification.

Limitations

This survey focuses on academic work in formal mathematical reasoning with LLMs, organized around datasets, formalizers, and provers; it omits adjacent areas and industrial systems, relies on benchmark-driven evaluations, and adopts a taxonomy that involves some subjective design choices.

Acknowledgments

We acknowledge the use of generative AI tools in the preparation of this manuscript. Specifically, generative AI tools (especially Ai2Asta and ChatGPT) were used to assist with literature search and to help polish draft text. All outputs were critically reviewed and edited by the authors to ensure accuracy and integrity.

References

Tudor Achim, Alex Best, Alberto Bietti, Kevin Der, Mathïs Fédérico, Sergei Gukov, Daniel Halpern-Leistner, Kirsten Henningsgard, Yury Kudryashov, Alexander Meiburg, Martin Michelsen, Riley Patterson, Eric Rodriguez, Laura Scharff, Vikram Shanker, Vladmir Sicca, Hari Sowrirajan, Aidan Swope, Matyas Tamas, and 4 others. 2025. Aristotle: IMO-level automated theorem proving. *Preprint*, arXiv:2510.01346.

Zhangir Azerbayev, Hailey Schoelkopf, Keiran Paster, Marco Dos Santos, Stephen McAleer, Albert Q. Jiang, Jia Deng, Stanislas Polu, and 1 others. 2023. ProofNet: Autoformalizing and formally proving undergraduate theorems. *arXiv preprint arXiv:2310.07695*.

Kaito Baba, Chaoran Liu, Shuhei Kurita, and Akiyoshi Sannai. 2025. Prover agent: An agent-based framework for formal mathematical proofs. *ArXiv*, abs/2506.19923.

Guoxin Chen, Jing Wu, Xinjie Chen, Wayne Xin Zhao, Ruihua Song, Chengxi Li, Kai Fan, Dayiheng Liu, and Minpeng Liao. 2025a. Reform: Reflective autoformalization with prospective bounded sequence optimization. *ArXiv*, abs/2510.24592.

Jiangjie Chen, Wenxiang Chen, Jiacheng Du, Jinyi Hu, Zhicheng Jiang, Allan Jie, Xiaoran Jin, Xing Jin, Chenggang Li, Wenlei Shi, Zhihong Wang, Mingxuan Wang, Chenrui Wei, Shufa Wei, Huajian Xin, Fan Yang, Weihao Gao, Zheng Yuan, Tianyang Zhan, and 3 others. 2025b. Seed-prover 1.5: Mastering undergraduate-level theorem proving via learning from experience. *Preprint*, arXiv:2512.17260.

Luoxin Chen, Jinming Gu, Liankai Huang, Wenhao Huang, Zhicheng Jiang, Allan Jie, Xiaoran Jin, Xing Jin, Chenggang Li, Kaijing Ma, Cheng Ren, Jiawei Shen, Wenlei Shi, Tong Sun, He Sun, Jiahui Wang, Siran Wang, Zhihong Wang, Chenrui Wei, and 17 others. 2025c. Seed-prover: Deep and broad reasoning for automated theorem proving. *Preprint*, arXiv:2507.23726.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. 2021. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*.

Leonardo de Moura, Soonho Kong, Jeremy Avigad, Floris van Doorn, and Jakob von Raumer. 2015. The Lean theorem prover (system description). In *Automated Deduction – CADE-25*, pages 378–388.

William Feng, Ethan Lou, and Aryan Sharma. 2026. Surface sensitivity in lean 4 autoformalization. *Preprint*, arXiv:2604.23135.

Emily First, Markus N. Rabe, Talia Ringer, and Yuriy Brun. 2023. Baldur: Whole-proof generation and repair with large language models. In *Proceedings of the 31st ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering*, pages 1229–1241.

Binxin Gao and Jingjun Han. 2025. Max it or miss it: Benchmarking llm on solving extremal problems. *ArXiv*, abs/2510.12997.

Bofei Gao, Feifan Song, Zhe Yang, Zefan Cai, Yibo Miao, Qingxiu Dong, Lei Li, Chenghao Ma, Liang Chen, Runxin Xu, Zhengyang Tang, Benyou Wang, Daoguang Zan, Shanghaoran Quan, Ge Zhang, Lei Sha, Yichang Zhang, Xuancheng Ren, Tianyu Liu, and Baobao Chang. 2024. Omni-math: A universal olympiad level mathematic benchmark for large language models. *ArXiv*, abs/2410.07985.

Guoxiong Gao, Yutong Wang, Jiedong Jiang, Qi Gao, Zihan Qin, Tianyi Xu, and Bin Dong. 2025. Herald: A natural language annotated lean 4 dataset. *Preprint*, arXiv:2410.10878.

Dadi Guo, Jiayu Liu, Zhiyuan Fan, Zhitao He, Haoran Li, Yumeng Wang, and Yi R. Fung. 2025a. Mathematical proof as a litmus test: Revealing failure modes of advanced large reasoning models. *ArXiv*, abs/2506.17114.

Qianyun Guo, Jianing Wang, Jianfei Zhang, Deyang Kong, Xiangzhou Huang, Xiangyu Xi, Wei Wang, Jingang Wang, Xunliang Cai, Shikun Zhang, and Wei Ye. 2025b. Autoformalizer with tool feedback. *ArXiv*, abs/2510.06857.

Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. 2021. Measuring mathematical problem solving with the math dataset. *NeurIPS*.

Yanxing Huang, Xinling Jin, Sijie Liang, Peng Li, and Yang Liu. 2025. Formarl: Enhancing autoformalization with no labeled data. *ArXiv*, abs/2508.18914.

Thomas Hubert, Rishi Mehta, Laurent Sartran, Miklós Z. Horváth, Goran Žužić, Eric Wieser, Aja Huang, Julian Schrittwieser, Yannick Schroecker, Hussain Masoom, Ottavia Bertolli, Tom Zahavy, Amol Mandhane, Jessica Yung, Iuliya Beloshapka, Borja Ibarz, Vivek Veeriah, Lei Yu, Oliver Nash, and 19 others. 2025. Olympiad-level formal mathematical reasoning with reinforcement learning. *Nature*.

Albert Qiaochu Jiang, S. Welleck, J. Zhou, Wenda Li, Jiacheng Liu, M. Jamnik, Timothée Lacroix, Yuhuai Wu, and Guillaume Lample. 2022. Draft, sketch, and prove: Guiding formal theorem provers with informal proofs. *ArXiv*, abs/2210.12283.

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

Haocheng Ju, Guoxiong Gao, Jiedong Jiang, Bin Wu, Zeming Sun, Leheng Chen, Yutong Wang, Yuefeng Wang, Zichen Wang, Wanyi He, Peihao Wu, Liang Xiao, Ruochuan Liu, Bryan Dai, and Bin Dong. 2026. Automated conjecture resolution with formal verification. *Preprint*, arXiv:2604.03789.

Takeshi Kojima, Shixiang Shane Gu, Machel Reid, Yutaka Matsuo, and Yusuke Iwasawa. 2022. Large language models are zero-shot reasoners. *arXiv preprint arXiv:2205.11916*.

Austin Letson, Leopoldo Sarra, Auguste Poiroux, Oliver Dressler, Paul Lezeau, Dhyan Aranha, Frederick Pu, Aaron Hill, Miguel Corredera Hidalgo, Julian Berman, George Tsoukalas, and Lenny Taelman. 2026. SorryDB: Can AI provers complete real-world lean theorems? *Preprint*, arXiv:2603.02668.

Wenda Li, Lei Yu, Yuhuai Wu, and Lawrence Charles Paulson. 2021. Isarstep: a benchmark for high-level mathematical reasoning. In *International Conference on Learning Representations*.

Yujia Lin, Shicheng Tang, Bohan Lyu, Jiawei Wu, Hanze Lin, Kaiyu Yang, Jia Li, Daren Xia, Danqi Chen, Sanjeev Arora, and Sean Welleck. 2025. Goedel-prover-v2: Scaling formal theorem proving with scaffolded data synthesis and self-correction. *Preprint*, arXiv:2508.03613.

Chengwu Liu, Jianhao Shen, Huajian Xin, Zhengying Liu, Ye Yuan, Haiming Wang, Wei Ju, Chuanyang Zheng, Yichun Yin, Lin Li, Ming Zhang, and Qun Liu. 2023. Fimo: A challenge formal dataset for automated theorem proving. *ArXiv*, abs/2309.04295.

Junqi Liu, Zhibin Lin, Zhibin Zhang, Zilong Wang, Yifei Chen, Zhenyu Wang, Yu Zhang, Yu Wang, Zhiheng Zhang, Yuxiao Wang, Zihan Zhang, Ziyi Zhang, Jing Li, Kai Wang, and Zhihong Zhang. 2025a. CombiBench: Benchmarking LLM capability for combinatorial mathematics. *arXiv preprint arXiv:2505.03171*. Lean 4 benchmark with 100 combinatorial problems from middle school to IMO/university level; includes Fine-Eval framework for proof and fill-in-the-blank evaluation.

Qi Liu, Xinhao Zheng, Xudong Lu, Qinxiang Cao, and Junchi Yan. 2025b. Rethinking and improving autoformalization: Towards a faithful metric and a dependency retrieval-based approach. In *The Thirteenth International Conference on Learning Representations*.

Xiaoyang Liu, Kangjie Bao, Jiashuo Zhang, Yunqi Liu, Yu Chen, Yuntian Liu, Yang Jiao, and Tao Luo. 2025c. Atlas: Autoformalizing theorems through lifting, augmentation, and synthesis of data. *ArXiv*, abs/2502.05567.

Wentao Long, Yunfei Zhang, Chenyi Li, Li Zhou, Chumin Sun, and Zaiwen Wen. 2026. Cam-bench: A benchmark for computational and applied mathematics in lean. *Preprint*, arXiv:2605.17255.

Norman Megill and David A. Wheeler. 2019. *Metamath: A Computer Language for Mathematical Proofs*. Lulu Press.

Logan Murphy, Kaiyu Yang, Jialiang Sun, Zhaoyu Li, Anima Anandkumar, and Xujie Si. 2024. Autoformalizing euclidean geometry. *ArXiv*, abs/2405.17216.

Tobias Nipkow, Lawrence C. Paulson, and Markus Wenzel. 2002. *Isabelle/HOL: A Proof Assistant for Higher-Order Logic*, volume 2283 of *Lecture Notes in Computer Science*. Springer. Latest Isabelle release: Isabelle2024 (2024).

Nilay Patel, Noah Arias, Davit Babayan, Victoria Cochran, Timothy Libman, Hafsah Mahmood, Liam McCarty, Soli Munoz, Laurel Willey, and Jeffrey Flanigan. 2026. Mathatlas: A benchmark for autoformalization in the wild. *Preprint*, arXiv:2605.14061.

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. 2025. Criticlean: Critic-guided reinforcement learning for mathematical formalization. *Preprint*, arXiv:2507.06181.

Z. Ren, Zhihong Shao, Jun-Mei Song, Huajian Xin, Haocheng Wang, Wanjia Zhao, Liyue Zhang, Zhe Fu, Qihao Zhu, Dejian Yang, Z. F. Wu, Zhibin Gou, Shirong Ma, Hongxuan Tang, Yuxuan Liu, Wenjun Gao, Daya Guo, and Chong Ruan. 2025. Deepseek-prover-v2: Advancing formal mathematical reasoning via reinforcement learning for subgoal decomposition. *ArXiv*, abs/2504.21801.

Borja Requena Pozo, Austin Letson, Krystian Nowakowski, Izan Beltran Ferreiro, and Leopoldo Sarra. 2026. A minimal agent for automated theorem proving. *Preprint*, arXiv:2602.24273.

Ziju Shen, N. Huang, Fanyi Yang, Yutong Wang, Guoxiong Gao, Tianyi Xu, Jiedong Jiang, Wanyi He, Pu Yang, Mengzhou Sun, Haocheng Ju, Peihao Wu, Bryan Dai, and Bin Dong. 2025. Real-prover: Retrieval augmented lean prover for mathematical reasoning. *ArXiv*, abs/2505.20613.

Alexander K. Taylor, Junyi Zhang, Ethan Ji, Vigyan Sahai, Haikang Deng, Yuanzhou Chen, Yifan Yuan, Di Wu, Jia-Chen Gu, Kai-Wei Chang, Nanyun Peng, Amit Sahai, and Wei Wang. 2026. Taobench: Do

automated theorem prover llms generalize beyond mathlib? *Preprint*, arXiv:2603.12744.

Amitayush Thakur, George Tsoukalas, Yeming Wen, Jimmy Xin, and Swarat Chaudhuri. 2023. An in-context learning agent for formal theorem-proving. *Preprint*, arXiv:2310.04353.

The Coq Development Team. 1999–. *The Coq Proof Assistant Reference Manual*. INRIA. Version 8.19 (2024), Version 8.20 expected 2025.

G. Tsoukalas, Jasper Lee, J. Jennings, Jimmy Xin, Michelle Ding, Michael Jennings, Amitayush Thakur, and Swarat Chaudhuri. 2024. Putnambench: Evaluating neural theorem-provers on the putnam mathematical competition. *ArXiv*, abs/2407.11214.

Sumanth Varambally, Thomas Voice, Yanchao Sun, Zhifeng Chen, Rose Yu, and Ke Ye. 2025. Hilbert: Recursively building formal proofs with informal reasoning. *ArXiv*, abs/2509.22819.

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, and 21 others. 2025a. Kimina-prover preview: Towards large formal reasoning models with reinforcement learning. *Preprint*, arXiv:2504.11354.

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

Shaoqi Wang, Lu Yu, and Chunjie Yang. 2025c. Improving autoformalization using direct dependency retrieval. *Preprint*, arXiv:2511.11990.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V. Le, and Denny Zhou. 2022. Chain-of-thought prompting elicits reasoning in large language models. *arXiv preprint arXiv:2201.11903*.

Nicolas Wischermann, C. M. Verdun, Gabriel Poesia, and Francesco Noseda. 2025. Proofcompass: Enhancing specialized provers with llm guidance. *ArXiv*, abs/2507.14335.

Yuhuai Wu, Sean Welleck, Denny Zhou, Xinyun Li, Mateja Jamnik, Thomas Lacroix, Ilya Sutskever, and Percy Liang. 2022. Autoformalization with large language models. *arXiv preprint arXiv:2205.12615*. Foundational work on LLM-based autoformalization, aligning with the original 'Kimina-Autoformalizer' concept.

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

Zijian Wu, Suozhi Huang, Zhejian Zhou, Huaiyuan Ying, Wenwei Zhang, Dahua Lin, and Kai Chen. 2024. Internlm2.5-stepprover: Advancing automated theorem proving via critic-guided search. In *unknown*.

Ran Xin, Chenguang Xi, Jie Yang, Feng Chen, Hang Wu, Xia Xiao, Yifan Sun, Shen Zheng, and Kai Shen. 2025. Bfs-prover: Scalable best-first tree search for llm-based automatic theorem proving. In *Annual Meeting of the Association for Computational Linguistics*.

Huaiyuan Ying, Zijian Wu, Yihan Geng, JIayu Wang, Dahua Lin, and Kai Chen. 2024. Lean workbook: A large-scale lean problem set formalized from natural language math problems. In *The Thirty-eight Conference on Neural Information Processing Systems Datasets and Benchmarks Track*.

Zhouliang Yu, Ruotian Peng, Keyi Ding, Yizhe Li, Zhongyuan Peng, Minghao Liu, Yifan Zhang, Zheng Yuan, Huajian Xin, Wenhao Huang, Yandong Wen, and Weiyang Liu. 2025. FormalMATH: Benchmarking formal mathematical reasoning of large language models. *arXiv preprint arXiv:2505.02735*. Lean 4 benchmark with 5,560 verified problems across algebra, calculus, number theory, etc.; includes autoformalization pipeline for scalability.

Lan Zhang, Xin Quan, and André Freitas. 2024. Consistent autoformalization for constructing mathematical libraries. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*.

Lan Zhang, Marco Valentino, and Andre Freitas. 2025a. Autoformalization in the wild: Assessing LLMs on real-world mathematical definitions. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pages 1720–1738, Suzhou, China. Association for Computational Linguistics.

Lan Zhang, Marco Valentino, and Andre Freitas. 2025b. Masa: Llm-driven multi-agent systems for autoformalization. *ArXiv*, abs/2510.08988.

Lan Zhang, Marco Valentino, and André Freitas. 2026. Monotonic reference-free refinement for autoformalization. *Preprint*, arXiv:2601.23166.

Meiru Zhang, Philipp Borchert, Milan Gritta, and Gerasimos Lampouras. 2025c. Drift: Decompose, retrieve, illustrate, then formalize theorems. *ArXiv*, abs/2510.10815.

Michihiro Zhang, Ofir Press, and Yejin Wu. 2023. Analogical prompting. *arXiv preprint arXiv:2305.13574*.

Kunhao Zheng, Jesse Michael Han, and Stanislas Polu. 2021. minif2f: A cross-system benchmark for formal olympiad-level mathematics. *arXiv preprint arXiv:2109.04388*.

Kuo Zhou and Lu Zhang. 2025. Step-wise formal verification for llm-based mathematical problem solving. *ArXiv*, abs/2505.20869.

Appendix A: Metric Glossary

Evaluation in formal mathematical reasoning is difficult because formalizers and provers are often compared under different task boundaries, proof-assistant environments, and resource budgets. We summarize the most common metrics used in the surveyed literature.

Compilation rate. Compilation rate measures the fraction of generated formal statements or proofs that are accepted by the target proof assistant. Given a set of N generated outputs $\{y_i\}_{i=1}^{N}$ and a verifier $\mathcal{V}$, where $\mathcal{V}(y_i) = 1$ if y_i compiles and $\mathcal{V}(y_i) = 0$ otherwise, compilation rate is

$$\mathrm{CompRate} = \frac{1}{N}\sum_{i=1}^{N}\mathcal{V}(y_i).$$

This metric is necessary because non-compiling outputs cannot be formally checked. However, it is not sufficient for semantic correctness: a statement may type-check while omitting assumptions, changing quantifier scope, selecting an unintended definition, or formalizing a theorem different from the intended informal statement.

Semantic equivalence. Semantic equivalence measures whether a generated formal statement preserves the meaning of a reference or intended formal statement. Let y be a generated formalization and $y^\star$ be the reference statement. Ideally, semantic equivalence requires that y and $y^\star$ have the same mathematical content, even if they use different but equivalent formal expressions. In practice, this is difficult because proof assistants may represent equivalent concepts through different definitions, coercions, or library lemmas.

Bidirectional Extended Definitional Equivalence. BEq, or Bidirectional Extended Definitional Equivalence, was proposed as a neuro-symbolic metric for statement-level autoformalization (Liu et al., 2025b). Informally, two formal statements y and $y^\star$ are BEq-equivalent if each can be derived from the other using semantics-preserving transformations, such as unfolding definitions, rewriting by definitional equalities, renaming variables, or applying simple equivalence-preserving proof steps. We can write this as

$$\mathrm{BEq}(y, y^\star) = 1 \quad \Longleftrightarrow \quad (y \Rightarrow y^\star) \wedge (y^\star \Rightarrow y),$$

where each implication is checked within a restricted proof/search procedure intended to preserve meaning rather than solve the full theorem from scratch. BEq is therefore stricter than compilation rate but more flexible than exact string or syntax-tree matching.

BEq@k. When a formalizer generates k candidate formalizations $\{y_1, \ldots, y_k\}$ for a reference statement $y^\star$, BEq@k measures whether at least one candidate is semantically equivalent to the reference:

$$\mathrm{BEq@}k = I\left[\exists i \in \{1, \ldots, k\} | \mathrm{BEq}(y_i, y^\star) = 1\right].$$

Over a dataset of N examples, the reported BEq@k score is the average of this indicator across all examples. Like pass@k, BEq@k is sensitive to the sampling budget k and should therefore be reported together with the number of generated candidates and the selection or filtering procedure.

Pass@k or success@k. Pass@k measures whether a prover solves a problem within k generated proof attempts. For a problem with generated attempts $\{p_1, \ldots, p_k\}$ and verifier $\mathcal{V}$, the per-problem pass@k indicator is

$$\mathrm{Pass@}k = I\left[\exists i \in \{1, \ldots, k\} | \mathcal{V}(p_i) = 1\right].$$

The dataset-level score averages this value across all problems. This metric is useful for proof synthesis, but it is highly sensitive to the number of samples, timeout, search strategy, and number of proof-assistant calls. A high pass@k score may therefore reflect large test-time search rather than strong single-attempt reasoning.

Search budget. Search budget refers to the computational resources allowed during proof search. This may include the number of generated candidates, timeout per problem, number of proof-assistant calls, maximum search depth, number of refinement rounds, and degree of parallelism. Search budget should be reported because repeated interaction with the proof assistant can substantially increase success rates. Without budget normalization, cross-paper comparisons between prover systems can be misleading.

Library snapshot. A library snapshot specifies the exact proof-assistant version and formal-library version used during evaluation. This is crucial for reproducibility because libraries such as Mathlib evolve rapidly: theorem names, definitions, APIs, and available lemmas may change over time. A proof that succeeds under one snapshot may fail under another, and retrieval-based systems are especially sensitive to such changes.

Proof length. Proof length measures the size of a successful proof, usually in terms of tactic steps, proof-term size, or generated tokens. Although shorter proofs are not always mathematically better, proof length helps characterize search efficiency, dependence on automation, and readability. It is also useful for distinguishing systems that solve problems through concise structured reasoning from those that rely on long generated tactic scripts.

7 Appendix B: Major Datasets for LLM-based Mathematical Reasoning

Table 1: Major datasets for LLM-based mathematical reasoning (informal to formal spectrum)

Dataset	# Problems / Statements	Language / Style	Main Domains	Key Features
GSM8K	8,500	Natural language	Grade-school arithmetic word problems	First chain-of-thought benchmark
MATH	12,500	Natural language	Pre-algebra, algebra, geometry, number theory, counting	Full solutions, difficulty levels 1–5
OmniMATH	4,617	Natural language	Algebra, geometry, number theory, combinatorics	Multilingual Olympiad problems
MiniF2F	488	Lean, Metamath, Isabelle	AMC/AIME/IMO + high-school problems	First multi-prover formal benchmark
ProofNet	3,675	Lean 3/4 (human-readable)	Algebra, analysis, topology, number theory, combinatorics, logic	Clean textbook-style proofs, ProofNet-Hard subset
FIMO	843	Lean 4	Algebra, geometry, number theory, combinatorics	IMO + national Olympiad problems
PutnamBench	131	Lean 4	Analysis, algebra, number theory, combinatorics, topology	Putnam competition problems formalized in Lean
IsarStep	2,018	Isabelle/Isar	Set theory, lattices, number theory, analysis	Human-like high-level proof steps
LeanWorkbook	23,422	Lean 4	Linear algebra, analysis, topology, etc.	Undergraduate textbook exercises
RFMDataset	200	Natural language + formal	Combinatorics, analysis, number theory, logic	Failure-mode diagnostic dataset
HEARALD	1,200,000+ tactics	Lean 4 + NL commentary	All Mathlib domains	Massive tactic–comment alignment corpus
FormalMATH	5,560	Lean 4	College algebra, analysis, differential equations, probability, number theory, graph theory	Human-in-the-loop autoformalization
CombiBench	100	Lean 4	Combinatorics (counting, graph, extremal, Ramsey)	Fine-grained evaluation with Fine-Eval scoring
ExtremBench	93	Natural language / answer-based extremal reasoning	Inequality-based optimization problems	Tests maxima/minima reasoning under constraints
Def_Wiki	56	Formal definitions	Wikipedia domains (groups, rings, topology, measure, etc.)	Real-world ambiguous definitions
Def_ArXiv	30	Formal definitions	arXiv papers (category theory, functional analysis, etc.)	Research-level context grounding
FineLeanCorpus	285,957	Lean 4 + NL statements	Algebra, analysis, number theory, combinatorics, topology, geometry	Largest verified autoformalization corpus (CriticLean)
MathAtlas	$\sim$52,000 objects	Informal + dependency-rich formalization targets	Graduate-level algebra, analysis, topology, geometry, etc.	In-the-wild autoformalization with contextual dependencies
TaoBench	150 exercises	Lean 4	Real analysis from first principles	Tests robustness beyond standard Mathlib definitions
CAM-Bench	1,000 proof targets	Lean 4	Optimization, numerical linear algebra, numerical analysis	Computational/applied mathematics with recovered textbook context