Mathematical Scientific Discovery Using Large Language Models: A Systematic Literature Review

Vinita Gangaram Jansari

v.jansar@clemson.edu

Clemson University

Carlos Toxtli Hernandez

Clemson University

Michael Burr

Clemson University

Austin LaHue

Clemson University

Ben Nye

Colorado College

Luis David Garcia Puente

Colorado College

Research Article

Keywords: Large Language Models, Mathematical Discovery, Theorem Generation, Conjecture Generation, Formal Verification, PRISMA, Systematic Review, Automated Theorem Proving, Symbolic Regression

Posted Date: August 3rd, 2026

DOI: https://doi.org/10.21203/rs.3.rs-10387804/v1

Additional Declarations: No competing interests reported.

Mathematical Scientific Discovery Using Large Language Models: A Systematic Literature Review

Vinita Gangaram Jansari[1*†], Carlos Toxtli Hernandez[2†], Michael Burr[3], Austin LaHue[2], Ben Nye[4], Luis David Garcia Puente[4]

[1*]School of Mechanical and Automotive Engineering, Clemson University, Greenville, SC, USA.
[2]School of Computing, Clemson University, Clemson, SC, USA.
[3]School of Mathematical and Statistical Sciences, Clemson University, Clemson, SC, USA.
[4]School of Mathematical and Computer Science, Colorado College, Colorado Springs, CO, USA.

*Corresponding author(s). E-mail(s): vjansar@clemson.edu;
Contributing authors: ctoxtli@clemson.edu; burr2@clemson.edu; jlahue@clemson.edu; bnye@ColoradoCollege.edu; lgarciapuente@coloradocollege.edu;
†These authors contributed equally to this work.

Abstract

The intersection of large language models (LLMs) and mathematical discovery represents a rapidly evolving frontier in artificial intelligence research. This systematic literature review, conducted following the Preferred Reporting Items for Systematic Reviews and Meta-Analyses (PRISMA) 2020 guidelines, examines the current state of research on using LLMs for generating new mathematical knowledge, including conjectures, theorems, lemmas, axioms, and counterexamples. From an initial pool of 63 records published through December 31, 2025, we identified 32 studies that met our inclusion criteria after rigorous screening. Our analysis reveals seven distinct technical paradigms: evolutionary search methods, reinforcement learning (RL) approaches, self-play and expert iteration, neuro-symbolic hybrid systems, in-context learning pipelines, specialized pretraining objectives, and equation discovery techniques. We amalgamate findings across formal verification systems (including Lean, Isabelle, PVS, and Metamath), along with datasets, evaluation metrics, and architectural choices. Our review

identifies that evolutionary approaches combined with formal verification have achieved the most significant mathematical discoveries, including improvements to Strassen's matrix multiplication algorithm and solutions to open problems in extremal combinatorics. We provide a technical taxonomy to guide practitioners in designing novel architectures for mathematical discovery, highlighting both proven strategies and open challenges in this emerging field.

Keywords: Large Language Models, Mathematical Discovery, Theorem Generation, Conjecture Generation, Formal Verification, PRISMA, Systematic Review, Automated Theorem Proving, Symbolic Regression

1 Introduction

The automation of mathematical discovery has been a long-standing goal in artificial intelligence, dating back to early symbolic AI systems and automated theorem provers [1]. Recent advances in large language models (LLMs) have reinvigorated this pursuit, demonstrating unprecedented capabilities in mathematical reasoning, proof generation, and crucially, the generation of novel mathematical knowledge.

Unlike traditional automated theorem proving, which focuses on verifying or proving existing mathematical statements, mathematical *discovery* involves the creation of new mathematical statements: conjectures that propose previously unknown relationships, theorems that establish new truths, lemmas that serve as stepping-stones to deeper results, and counterexamples that disprove false conjectures. This creative aspect of mathematics has historically been considered uniquely human, requiring intuition, pattern recognition, deep domain expertise, and inspiration.

The emergence of LLMs trained on large scale mathematical corpora has significantly reshaped the landscape of automated reasoning and mathematical statement discovery. Systems such as FunSearch [2] have discovered new constructions in extremal combinatorics that surpass the best known human results. AlphaTensor [3] discovered faster matrix multiplication algorithms, improving upon Strassen's seminal 1969 algorithm for the first time in more than 50 years. These achievements demonstrate that AI systems can not only assist mathematicians, but also independently contribute to mathematical knowledge.

1.1 Motivation and Scope

This systematic review addresses a critical need in the research community: a comprehensive summary of methods, architectures, and results in LLM-based mathematical discovery. The rapid pace of development in this field, with many papers appearing in 2024-2025, makes it essential to consolidate knowledge and identify effective technical approaches.

Our review specifically focuses on **knowledge generation** rather than **problem solving**. In this context, problem solving uses LLMs to solve existing problems, prove known theorems, or answer mathematical questions, and it represents applications of existing knowledge. On the other hand, mathematical discovery uses LLMs

to generate *new* mathematical objects, conjectures, theorems, lemmas, counterexamples, algorithms, or constructions, that contribute to mathematical knowledge. This distinction separates mathematical problem solving from mathematical discovery and is crucial for practitioners building LLM systems aimed at advancing mathematical research rather than merely assisting with more routine mathematical tasks.

1.2 Research Questions

This systematic review addresses five research questions. First, we investigate what technical architectures and methodologies have been proposed for LLM-based mathematical discovery (RQ1). Second, we examine which formal verification systems are used to validate generated mathematical knowledge (RQ2). Third, we explore what datasets and benchmarks exist for training and evaluating mathematical discovery systems (RQ3). Fourth, we analyze the key success factors and limitations of current approaches (RQ4). Finally, we survey which mathematical domains and problem types have been addressed (RQ5).

1.3 Contributions

This review makes five principal contributions. We provide a rigorous Preferred Reporting Items for Systematic Reviews and Meta-Analyses (PRISMA)-compliant [4] systematic review of 32 studies on LLM-based mathematical discovery published through December 2025. We develop a technical taxonomy of seven distinct paradigms for mathematical knowledge generation. We offer a comprehensive synthesis of formal verification systems, datasets, and evaluation approaches. We present actionable guidance for practitioners designing mathematical discovery systems. Finally, we identify open challenges and promising research directions for this emerging field.

2 Background and Definitions

Before presenting our methodology, we provide definitions of key terms and systems referenced throughout this review.

2.1 Key Concepts

A **large language model (LLM)** is a neural network trained on large text corpora using self-supervised learning, typically based on the transformer architecture. In this review, we include both general-purpose LLMs, such as GPT-4 [5] and Claude [6], as well as code-specialized variants, including Codex [7] and StarCoder [8]. **Mathematical discovery** refers to the generation of new mathematical knowledge, including conjectures, theorems, lemmas, algorithms, counterexamples, or constructions that were previously unknown. **Formal verification** involves the use of proof assistants such as Lean [9], Isabelle [10], or Coq [11] to mechanically verify the correctness of mathematical statements and proofs.

A **conjecture** is a mathematical statement believed to be true but not yet proven, while **theorem and lemma generation** refer to the automatic creation of new

mathematical statements along with their proofs. A **counterexample** is a specific instance that disproves a mathematical statement, demonstrating its falsity.

Symbolic regression is the task of discovering mathematical equations from data, expressing relationships in symbolic form. **Expert iteration** is a training procedure that alternates between generating solutions and training on the successful solutions. **Evolutionary search** refers to optimization through population-based methods inspired by biological evolution, including mutation, selection, and recombination.

2.2 Formal Verification Systems

The primary formal verification systems referenced in this review include Lean [9], Isabelle [10], Metamath [12], and PVS [13]. Lean is a proof assistant developed at Microsoft Research, with Lean 4 being the current version; its Mathlib library contains more than one million lines of formalized mathematics. Isabelle is a generic proof assistant supporting various logics, particularly higher-order logic (HOL), and its archive of formal proofs contains over 700 entries of formalized mathematics. Metamath is a minimalist formal system with a large library of proofs in set theory and related areas. The prototype verification system (PVS) is a proof assistant that provides support for decision procedures.

3 Methods

This systematic review follows the Preferred Reporting Items for Systematic Reviews and Meta-Analyses (PRISMA) 2020 guidelines [14]. We report our methodology transparently to ensure reproducibility. The literature search was conducted between January 15-20, 2026, covering publications through December 31, 2025. This review was not pre-registered due to the exploratory nature of summarizing a rapidly evolving field; however, all screening decisions and extracted data are available in the supplementary materials.

3.1 Eligibility Criteria

Studies were included if they met all five inclusion criteria.

I1: Domain: The study must address mathematical discovery, including conjecture generation, theorem generation, lemma generation, counterexample generation, algorithm discovery, or construction finding.

I2: Technology: The study must employ LLMs, neural language models, or transformer-based architectures as a core component of the discovery pipeline.

I3: Novelty Generation: The study must demonstrate or propose methods for generating new mathematical knowledge, not merely solving existing problems or proving known theorems.

I4: Focus: The primary application domain must be mathematics, including theoretical computer science, combinatorics, number theory, algebra, geometry, analysis, and mathematical physics.

I5: Type: The study must be a peer-reviewed publication, preprint on a recognized repository such as arXiv or OpenReview, or published in conference proceedings.

Studies were excluded if they met any of six exclusion criteria.

E1: Problem Solving Only: Focusing exclusively on solving existing mathematical problems without generating new mathematical statements.
E2: Non-Mathematical Domains: Applying LLMs to scientific discovery in non-mathematical domains such as biology, chemistry, or materials science without mathematical knowledge generation.
E3: No LLM Component Using purely symbolic methods, classical machine learning, or non-language-model neural networks without LLM integration.
E4: Duplicate Publications: Being near-duplicates of included papers.
E5: Non-English: Not being available in English.
E6: Insufficient Detail: Providing insufficient methodological detail to assess their approach to mathematical discovery.

3.2 Information Sources and Search Strategy

Records were obtained from a curated literature search encompassing major academic databases and preprint repositories, including Semantic Scholar, arXiv (specifically the cs.AI, cs.LG, cs.CL, and math.* categories), OpenReview (including ICLR, NeurIPS, and ICML submissions), JMLR, Nature and Nature-affiliated journals (including Machine Intelligence), and the ACL Anthology. The search was conducted using keyword combinations including "large language model" AND ("theorem generation" OR "conjecture" OR "mathematical discovery" OR "automated theorem" OR "lemma generation" OR "counterexample").

3.3 Selection Process

The selection process followed a two-stage screening procedure. In the first stage of title and abstract screening, all records were screened based on their titles and abstracts to assess potential relevance, with records clearly outside the scope being excluded. In the second stage of full-text assessment, remaining records underwent full-text review to verify eligibility against all inclusion and exclusion criteria. Screening was performed through careful semantic analysis of each paper's abstract, focusing on whether the study addresses mathematical knowledge generation rather than mere problem solving.

3.4 Data Extraction

For each included study, we extracted bibliographic information including authors, year, and venue, the technical approach and architecture employed, the LLM models used, the formal verification systems employed, the datasets and benchmarks used, the types of mathematical objects generated, the mathematical domains studied, the key results and discoveries, and the evaluation metrics and performance measures.

3.5 Risk of Bias Assessment

Given the nascent nature of this field and the diversity of study designs, we did not conduct a formal risk of bias assessment using standardized tools. Instead, we note methodological considerations in our synthesis, where relevant, acknowledging this as a limitation.

4 Results

This section reports on the study selection process and the study characteristics of our systematic review. We summarize the study selection process, including the records identified, screened, excluded, and retained for inclusion. We then provide basic data on the distribution of records retained for our study.

4.1 Study Selection

Figure 1 presents the PRISMA flow diagram summarizing our study selection process. From the 63 records initially identified from database searches, we removed 2 duplicates, leaving 61 records for screening. After title and abstract screening and full-text assessment, 29 records were excluded for not meeting inclusion criteria, resulting in 32 studies included in the final review.

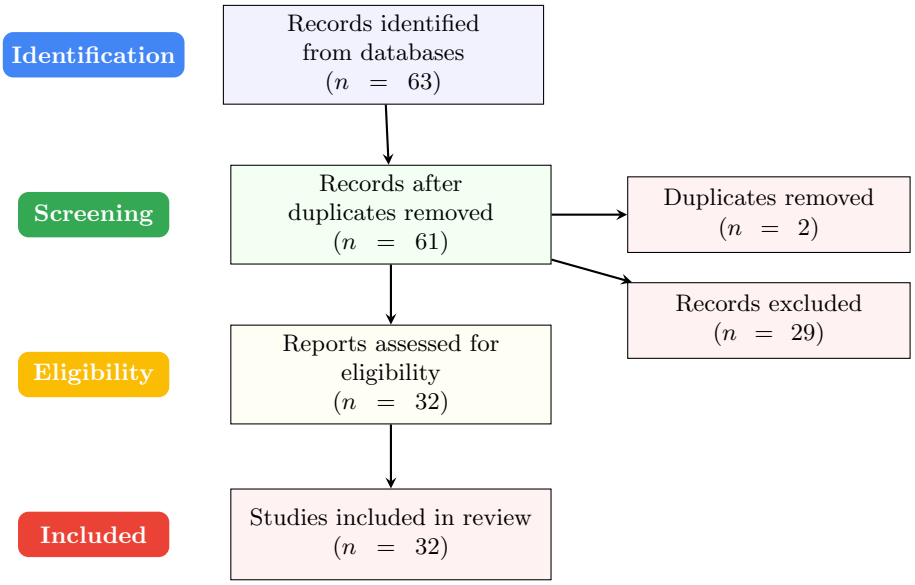

Fig. 1: PRISMA 2020 flow diagram showing the study selection process.

The 29 excluded records were removed for the following reasons: seven studies focused on problem solving rather than generating new knowledge (E1), twelve studies addressed non-mathematical domains such as biology, chemistry, materials science,

or finance (E2), four studies used purely symbolic or classical machine learning methods without LLM integration (E3), four studies represented general AI or reasoning frameworks without mathematical discovery focus (E4), and one study was a duplicate publication of an included paper. Ine additional study from 2026 fell outside our temporal scope.

4.2 Study Characteristics

The included studies span 2020-2025, with a marked increase in recent years reflecting the rapid growth of this field: two studies from 2020, three from 2021, one from 2022, three from 2023, seven from 2024, and sixteen from 2025. The studies appeared in diverse venues, with four papers published in the Nature family of journals (including FunSearch [2], AlphaTensor [3], the Ramanujan Machine [15], and the Davies *et al.* [16] knot theory work), five papers in major machine learning conferences (ICLR, ICML, NeurIPS workshops, and NAACL), twenty one in arXiv preprints primarily in the cs.AI, cs.LG, and math.CO categories, and two in other venues including CEUR-WS and Physics of Fluids.

5 Technical Analysis of Discovery Paradigms

Based on our analysis, we identify the following seven distinct technical paradigms for LLM-based mathematical discovery. This section provides a detailed technical analysis of each paradigm, discussing implementation details, architectural choices, and results across studies.

5.1 Evolutionary Search Methods

Evolutionary search has emerged as one of the most powerful and scalable paradigms for autonomous mathematical discovery, driving breakthroughs in long-standing open problems across combinatorics, geometry, and matrix algebra. Rather than navigating raw mathematical solution spaces directly, modern evolutionary frameworks shift the search to the space of functional, executable programs, using LLMs as highly adaptive mutation and crossover operators [2]. These systems typically embed LLM samplers within closed-loop architectures that iteratively execute candidate code, measure performance via automated verifiers, and maintain dynamic databases of high-fitness programs [2, 17]. Recent iterations have expanded this frontier by introducing multi-LLM orchestration, sample-efficient selection mechanics, and continual learning to systematically explore dozens of theoretical problems at scale [18–20]. The following sections detail how these evolutionary coding agents have discovered novel lower bounds for the cap set and kissing number problems, optimized complex matrix multiplication, and achieved state-of-the-art geometric packing solutions.

5.1.1 FunSearch: Program Search with LLMs

FunSearch [2] introduced the paradigm of searching in the space of programs rather than solutions. The architecture pairs a pretrained LLM with a systematic evaluator in an evolutionary loop consisting of four components. The process begins with a seed

population of simple or randomly generated Python functions, often designed to satisfy basic syntactic and type constraints. These initial programs are evaluated to establish baseline fitness scores. From there, a program database maintains the population of programs organized by fitness, an LLM sampler selects high-performing candidates from this database to prompt the LLM for improved variants, an evaluator executes the generated programs on test cases to compute their fitness scores, and a selection mechanism adds programs that exceed set fitness thresholds back into the database. Over successive iterations, this loop drives the discovery of increasingly sophisticated and high-performing mathematical programs.

FunSearch discovered new constructions for the cap set problem, achieving improved lower bounds on the cap set problem in $\mathbb{F}_3^n$, a central problem in extremal combinatorics. It also discovered new heuristics for online bin packing that outperform decades of human-designed algorithms. The system uses Codey, a code-specialized PaLM variant [21], with automated execution and timeout for evaluation, maintaining populations of thousands of programs with problem-specific fitness functions.

5.1.2 AlphaEvolve: Evolutionary Coding Agent

AlphaEvolve [17] extends the evolutionary approach with an autonomous multi-LLM pipeline. Its key innovations include multi-LLM orchestration with multiple LLMs having different specializations; continuous feedback, where real-time evaluator feedback guides evolution; and applications to diverse domains including both mathematical and computational-infrastructure problems.

Among its mathematical discoveries, AlphaEvolve found a procedure for multiplying two 4×4 complex matrices using 48 scalar multiplications, the first improvement over Strassen's algorithm (which requires 49 multiplications) in 56 years. The system also advanced the 11-dimensional kissing number problem by discovering a configuration of 593 tangent spheres, establishing a new lower bound in that dimension. In other areas, such as uncertainty inequalities and related extremal or geometric problems, AlphaEvolve tightened known bounds and recovered best-known constructions.

5.1.3 Mathematical Exploration at Scale

Georgiev *et al.* [18] applied AlphaEvolve to 67 open mathematical problems spanning analysis, combinatorics, geometry, and number theory. The system rediscovered the best-known solutions for most problems and discovered improved solutions for several problems. Notably, it demonstrated the ability to generalize finite results to general formulas and showed successful integration with AlphaProof [22] for proof generation.

5.1.4 ShinkaEvolve and CoEvo

ShinkaEvolve [19] addresses sample efficiency, meaning it is designed to find high-performing solutions using far fewer environment interactions and candidate-program evaluations than standard evolutionary approaches. Its technical innovations include parent sampling that balances exploration and exploitation using fitness-weighted selection, novelty-rejection sampling that avoids redundant exploration by rejecting similar programs, and a bandit-based LLM ensemble that

dynamically selects among multiple LLMs, based on performance. The system discovered a new state-of-the-art circle packing solution with only 150 samples, designed effective agentic harnesses for the mathematical reasoning section of the American Invitational Mathematics Examination (AIME), and provided an open-source implementation enabling broad adoption.

CoEvo [20] introduces continual learning to symbolic discovery with a dynamic knowledge library that continuously stores and retrieves discovered solutions, open-ended innovation that discovers new problems along with solutions, and multiple representations spanning natural language, mathematical expressions, and code.

5.2 Reinforcement Learning and Guided Search Approaches

Recent breakthroughs demonstrate that fundamental mathematical discovery, algorithm synthesis, and complex theorem generation can be effectively framed as sequential decision-making problems [3]. By structuring abstract mathematical search spaces as discrete game environments or combinatorial generation tasks, learning-driven systems can discover novel structures without relying on human-curated demonstration data [23, 24]. These approaches typically pair deep neural architectures with structured exploration mechanisms, such as tree search or cross-entropy policy optimization to navigate vast, non-linear search spaces [3, 23]. The following sections detail how these learning-guided search frameworks have successfully discovered novel matrix multiplication algorithms, disproved long-standing graph theory conjectures, and generated competition-level geometric problems.

5.2.1 AlphaTensor: Tensor Decomposition as a Game

AlphaTensor [3] formulates algorithm discovery as a single-player game. The state represents the current tensor (the remaining computation). The goal is to reduce the tensor to zero with as few actions as possible, where an action consists of selecting a rank-1 tensor to subtract. The architecture is based on AlphaZero, using Monte Carlo Tree Search for planning, a neural network for policy and value prediction, and training via self-play. The system discovered algorithms improving on Strassen for 4×4 matrices in finite fields, developed hardware-specific optimizations for practical efficiency, and produced provably correct algorithms via construction.

5.2.2 Deep Cross-Entropy for Combinatorial Constructions

Wagner [23] applied the deep cross-entropy method to find counterexamples by training neural networks to generate mathematical objects (e.g., graphs, matrices), with rewards based on proximity to counterexample criteria and cross-entropy method for policy optimization. The mathematical discoveries include counterexamples to the Brualdi-Cao conjecture on permanents, counterexamples to conjectures on graph eigenvalues, and multiple results in extremal combinatorics.

5.2.3 TongGeometry: Olympiad Problem Proposing

TongGeometry [24] integrates a guided tree search framework with LLM reasoning to both propose and solve advanced geometry problems. For problem generation, it enumerated over 6.7 billion geometry theorems involving auxiliary constructions. Among the problems it proposed, ten were submitted to major mathematical olympiads, with three selected for official competitions, including national team selection exams and top civil olympiads in China and the United States. For problem-solving, TongGeometry successfully solved all geometry problems in the IMO-AG-30 benchmark [24]. Its approach combines an efficient geometric reasoning engine for theorem enumeration, fine-tuned LLMs for guided search and verification, and symmetry-aware aesthetic evaluation to produce high-quality geometry problems.

5.3 Self-Play and Expert Iteration

Self-play and expert iteration have established a powerful bootstrapping paradigm for formal theorem proving, enabling automated systems to transcend the limits of static, human-annotated training data. At the core of this approach is a recursive optimization loop where models generate their own training signals by continuously alternating between problem generation and problem solving. By structuring this interaction as a dual-role framework, systems can dynamically calibrate an internal curriculum, forcing a prover to adapt as a conjecturer proposes increasingly challenging mathematical objectives [25]. Beyond individual proof trajectories, this iterative process allows agents to organically grow, retrieve, and evolve structured libraries of verified lemmas, creating a modular foundation of mathematical skills that can be recursively composed to tackle highly complex theorems [26]. The following sections explore how these self-correcting and knowledge-accumulating architectures have yielded dramatic performance gains across rigorous formal mathematical benchmarks such as miniF2F [27] and LeanWorkbook [28].

5.3.1 STP: Self-play Theorem Provers

STP [25] introduces a dual-role framework where a single model plays the role of both conjecturer and prover. The training dynamics involve conjecturer training to generate conjectures that are barely provable by the current prover (curriculum learning); prover training using standard expert iteration on successfully proven conjectures; and iterative refinement, where the conjecturer generates increasingly difficult problems as the prover improves. The system achieved a 26.3% success on LeanWorkbook, doubling the previous best of 13.2%, and state-of-the-art performance on the miniF2F-test with 61.1% at pass@3200, generating 19.8 billion tokens during training.

5.3.2 LEGO-Prover: Growing Skill Libraries

LEGO-Prover [26] maintains a growing library of verified lemmas through skill creation. The LLM proposes new lemmas during proving; retrieves skills, where it pulls relevant lemmas to help with the current proof; and evolves skills, where it refines existing lemmas to create useful variations. The system generated over 20,000 new

skills (theorems and lemmas), improved the pass rate on miniF2F-valid from 48.0% to 57.0%, and ablation studies showed that newly added skills improve success rate by 3.3 percentage points.

5.4 Neuro-Symbolic Hybrid Systems

Neuro-symbolic hybrid systems capitalize on the complementary strengths of neural intuition and symbolic rigor to automate complex mathematical reasoning. While deep neural networks excel at generating diverse, highly creative mathematical hypotheses and abstract structural schemas, their open-ended heuristic search lacks inherent guarantees of logical correctness [29]. To bridge this gap, neuro-symbolic architectures tightly couple LLMs with deterministic symbolic engines, template instantiators, and interactive theorem provers [29, 30]. Within this dual-layer paradigm, the neural component acts as a flexible generative engine that proposes high-level templates or raw conjectures, while the symbolic layer serves as an absolute logical filter, instantiating fine-grained details, searching for counterexamples, and formally certifying correctness [30]. The following sections explore how these hybrid frameworks use template-based schema generation and multi-stage verification pipelines to systematically discover valid mathematical lemmas and separate genuine theoretical insights from neural noise.

5.4.1 Lemmanaid: Template-Based Lemma Conjecturing

Lemmanaid [30] combines LLM generation with symbolic instantiation. The architecture involves template generation, where the LLM generates lemma templates (schemas with placeholders); symbolic instantiation, where symbolic methods fill in template details; and verification using the Isabelle proof assistant. The system discovered 29-39.5% of gold-standard human-written lemmas, representing an 8-15% improvement over neural-only methods, and was evaluated on the Isabelle HOL library [31] and the Archive of Formal Proofs [32].

5.4.2 Exploring Mathematical Conjecturing with GPT

Johansson and Smallbone [29] explored direct prompting of LLMs (GPT-3.5 and GPT-4) to generate mathematical conjectures, integrating them into a neuro-symbolic pipeline where symbolic theorem provers and counterexample finders verified or refuted the generated statements. Their experiments produced mixed results: while some conjectures were valid, many were not. This study showed that LLMs are good at proposing diverse and unexpected conjectures, but that this creativity comes with a high risk of error, so their outputs cannot be trusted on their own. Symbolic theorem provers and counterexample finders are therefore essential partners: they act as rigorous filters that test each LLM-generated conjecture, keeping only those that are logically sound and discarding the rest, which underscores that reliable mathematical discovery still depends on formal verification rather than model intuition.

5.5 In-Context Learning Pipelines

In-context learning (ICL) pipelines represent a highly flexible, inference-time paradigm for mathematical discovery, bypassing or augmenting traditional parameter updates by dynamically expanding the model's immediate prompt environment. Rather than relying solely on frozen static weights, these frameworks use stateful and iterative prompting loops where successfully verified theorems and emerging proof strategies are continuously accumulated and recycled into the context window [33]. By combining rule-based metadata extraction from established formal repositories like Mathlib with live execution feedback from proof assistants, these systems allow language models to bootstrap their localized reasoning capabilities in real time [33, 34]. Furthermore, anchoring these context-rich generation loops with tailored policy optimization strategies ensures that the resulting hypotheses remain non-trivial and mathematically rigorous [34]. The following sections examine how these iterative in-context architectures enable language models to navigate formal languages like Lean 4, rediscover research-level results, and systematically map complex university-level concepts in advanced topology.

5.5.1 Conjecturing-Proving Loop

Kasaura *et al.* [33] introduced the Conjecturing-Proving Loop, an iterative framework for discovering new theorems via in-context proof-learning in Lean 4. The pipeline uses an LLM to generate conjectures in the Lean format, attempts to prove them with the same LLM, guided by a Lean server, and adds successfully proved theorems and proofs to a growing library that is fed back as context for subsequent conjecturing and proving rounds. This context-enriched loop enables the LLM to discover and formally prove increasingly difficult theorems and to rediscover research-level results, including at least one theorem that could not be proved by the LLM without such contextual examples, even in natural language, thereby demonstrating the effectiveness of in-context proof learning for neural theorem proving

5.5.2 LeanConjecturer

LeanConjecturer [34] generates university-level conjectures (conjectures that are similar in difficulty and sophistication to the mathematics taught and studied in undergraduate-level university courses) through rule-based context extraction from Mathlib seed files, LLM-based theorem statement generation, iterative generation and evaluation, and GRPO (Group Relative Policy Optimization) for reinforcement learning (RL). From 40 seed files, the system produced 12,289 conjectures, with 3,776 identified as syntactically valid and non-trivial, and verified non-trivial theorems in topology including properties of semi-open, alpha-open, and pre-open sets.

5.6 Specialized Pretraining Objectives

While generic autoregressive language modeling excels at capturing the linear flow of natural language, standard next-token prediction often struggles to grasp the rigid, deeply nested hierarchical structure inherent in formal mathematics. To address this

limitation, specialized pretraining objectives restructure self-supervised learning to operate directly on the underlying syntax and mathematical rules of formal languages [35]. By shifting the pretraining paradigm from linear text completion to structural prediction, such as identifying or reconstructing missing subtrees within an Abstract Syntax Tree (AST), models can natively encode the multi-layered semantic and logical dependencies of higher-order logic [35]. This structure-aware foundation provides a robust inductive bias that drastically improves downstream performance on formal reasoning tasks, including type inference, premise selection, and equality completion. The following section explores how these tailored pretraining mechanisms optimize neural architectures to comprehend the fundamental, non-linear topology of formal proofs and mathematical statements.

5.6.1 Skip-Tree Training

Rabe *et al.* [35] introduced skip-tree training, a self-supervised pretraining objective tailored to formalized mathematics in HOL. Instead of standard next-token prediction, their method learns by predicting missing subtrees in the abstract syntax trees of formal expressions, enabling the model to capture rich structural and semantic dependencies in proofs and statements. This pretraining supports tasks such as type inference, suggesting missing premises or assumptions, completing equalities, and formulating conjecture-like continuations evaluated via provability. Models trained with skip-tree objectives achieve stronger performance than standard language models on a range of mathematical reasoning benchmarks, indicating the benefits of structure-aware pretraining for formal mathematics.

5.7 Equation Discovery

Equation discovery and symbolic regression aim to infer mathematical relationships from data. Traditional approaches rely on combinatorial searches over symbolic expressions, while newer methods use language models to guide the search space. Evaluation of these models remains challenging, as models may reproduce known formulas from training data. Recent benchmarking frameworks address this challenge by transforming known equations into alternative structures and introducing synthetic tasks to measure model performance across multiple domains [36]. The following section details these evaluation methods and baseline results.

5.7.1 LLM-SRBench

LLM-SRBench [36] provides a benchmark preventing memorization through the LSR-Transform benchmark, which transforms known equations into less common forms, and the LSR-Synth benchmark, which creates synthetic problems requiring data-driven reasoning. The benchmark contains 239 problems across four domains. The finding that the best methods achieve only 31.5% accuracy highlights the challenge of true equation discovery.

6 Formal Verification Systems

A distinguishing feature of mathematical discovery is the need for rigorous verification. Lean, including Lean 4, has emerged as the dominant verification system, used by nine of 32 studies (28.1%). Key advantages of Lean include Mathlib, its extensive library of formalized mathematics containing over one million lines; powerful proof automation through tactics; active development with Lean 4 representing modern language design; and strong community engagement from mathematicians and CS researchers.

Isabelle is used by two studies, offering HOL and an archive of formal proofs. Metamath is used by one study, providing minimal logic with a large library. PVS is also used by one study for its higher-order logic and decision procedures. Five studies use problem-specific evaluators rather than general-purpose proof assistants, as seen in FunSearch and AlphaEvolve. These offer faster evaluation and problem-tailored metrics, though with limited generalization and the need to verify correctness separately.

7 Datasets and Benchmarks

Prominent datasets and benchmarks that support mathematical discovery include Mathlib4 [37], containing over one million lines of Lean formalized mathematics from the community; LeanWorkbook [28], containing 57,000 problems for theorem proving evaluation used by STP [25]; miniF2F [27], containing 488 problems as a standard formal math benchmark used by multiple studies; LeanComb-Enhanced [38], containing 260,000 theorems for combinatorial identities generated by ATG4CI [38]; LeanNavigator [39], containing 4.7 million auto-generated Lean theorems with one billion tokens; ConjectureBench [40], an augmented dataset for conjecture evaluation; LLM-SRBench [36], containing 239 problems for equation discovery designed to prevent memorization; and the ATG Benchmark [41], a Metamath [12] split for theorem generation evaluation.

Researchers employ several strategies to generate training data. Human-curated datasets leverage existing formalized mathematics such as Mathlib and AFP [32]. Synthetic generation, exemplified by LeanNavigator with 4.7 million theorems and ATG4CI with 260,000 theorems, creates new training examples automatically. Symbolic mutation, as in counterexample generation through hypothesis removal, creates diverse training instances. Self-play, as in STP, has the conjecturer generating training data for the prover.

8 Mathematical Domains and Results

This section reviews the specific mathematical disciplines addressed by these computational frameworks and details their verified results. The literature is concentrated primarily in discrete fields, including combinatorics, graph theory, algebra, and number theory with applications in geometry, algorithm design, analysis, complexity theory, and topology.

8.1 Domains Addressed

The academic literature indicates a modest but growing presence across the mathematical sciences. **Combinatorics and graph theory** stands as the most heavily investigated domain (twelve papers), serving as a natural testing ground for discrete generation tasks such as optimizing cap sets, cracking online bin packing, and constructing counterexamples to complex graph eigenvalue conjectures. **Algebra and number theory** follows closely (eight papers), with studies focusing on fundamental physical constants, continued fractions, and group theory representation.

The spatial and structural domains of **geometry** (four papers) and **algorithm design** (four papers) have yielded massive theorem enumeration and significant computational acceleration, notably via hardware-tailored matrix multiplication and scheduling heuristics. Finally, the boundaries of continuous and structural mathematics are being actively pushed in **analysis and PDEs** (two papers) via empirical equation discovery, **complexity theory** (two papers) through the derivation of new inapproximability bounds for NP-hard problems, and **topology** (two papers) investigating semi-open sets and structural knot invariants.

8.2 Significant Mathematical Discoveries

The mathematical breakthroughs enabled by these neural and evolutionary systems are not merely incremental; several represent the resolution of decades-old bottlenecks. We highlight the most significant discoveries below:

Matrix Multiplication Optimization: Scaling up algorithmic efficiency, AlphaEvolve [17] achieved the first improvement over Strassen's landmark algorithm for multiplication of 4×4 complex matrices in 56 years, reducing the number of required operations from 49 to 48 scalar multiplications.

Extremal Combinatorics Bounds: FunSearch [2] successfully discovered novel, larger constructions for the cap set problem, establishing strictly improved lower bounds in $\mathbb{F}_3^n$ that surpassed longstanding human-curated limits.

Knot Theory Topology: Bridging distinct mathematical branches, Davies *et al.* [16] established a profound, previously unmapped connection between the algebraic and geometric structures of knots, resulting in a landmark publication in *Nature*.

Combinatorial Designs: Using the deep reasoning capabilities of advanced LLMs, the CPro1 framework successfully solved long-standing open instances for seven out of sixteen core problems documented in the authoritative *Handbook of Combinatorial Designs*.

Complexity Theory Limits: In theoretical computer science, AlphaEvolve expanded the boundaries of inapproximability, proving tight new hardness results and bounds for classical optimization challenges including MAX-3-CUT, MAX-4-CUT, and the metric Traveling Salesperson Problem.

Fundamental Constants: Operating in the realm of number theory, the Ramanujan Machine successfully conjectured and generated dozens of entirely new, provably correct continued fraction representations for fundamental mathematical constants including π and e.

Conjecture Refutation: Harnessing automated policy optimization, Wagner [23] disproved long-standing mathematical assumptions by generating explicit counterexamples to the Brualdi-Cao conjecture on permanents as well as several persistent graph eigenvalue conjectures.

Olympiad-Level Generation: Demonstrating high-level semantic and spatial synthesis, TongGeometry [24] achieved an unprecedented milestone in automated pedagogy and evaluation by having three of its original, system-proposed geometry problems selected for official deployment in highly competitive regional and national mathematical olympiads.

9 Synthesis: Key Patterns and Insights

This section analyzes the operational factors, structural limitations, and development trends observed across the reviewed mathematical discovery frameworks. Successful implementations typically integrate automated formal verification directly into the optimization loop, utilize iterative code or skill accumulation, and apply domain-specific structural representations. Primary limitations include high computational resource requirements, low generalizability across domains, verification execution bottlenecks, high volume of trivial outputs, and limited human interpretability of machine generated code and proofs.

9.1 Success Factors

A retrospective analysis of systems that have achieved verified mathematical breakthroughs highlights three critical, recurring success factors:

In-the-Loop Formal Verification: The most impactful systems avoid post-hoc verification, opting instead to tightly couple generative neural networks or evolutionary samplers with symbolic solvers and interactive proof assistants directly within the optimization loop [30, 33]. This closed-loop setup transforms verification from a passive filter into an active guide, converting rich semantic errors and interpreter failures into actionable feedback to steer subsequent generations.

Iterative Solution Accumulation and Self-Evolution: Rather than generating solutions in a single forward pass, premier discovery frameworks rely on recursive, self-improving dynamics. This is achieved either by evolving functional code bases over successive generations [2, 17], establishing competitive co-evolutionary curricula via conjecturer-prover self-play [25], or curating incrementally growing skill libraries of verified lemmas that can be composed to solve harder problems [26].

Exploitation of Deep Domain Structure: Universally, generalized token generation underperforms when compared to architectures that map abstract search spaces onto concrete, domain-specific mathematical structures. Highly successful

configurations explicitly frame their objectives through structured mathematical lenses, such as casting algorithm discovery as tensor decomposition [3], using symmetry-aware representations for geometry [24], or leveraging the recursive mathematical structure of continued fractions.

9.2 Common Limitations

Despite notable landmark discoveries, there are several systemic bottlenecks that restrict broad applicability:

Prohibitive Computational Overhead: High-fidelity mathematical discovery remains exceptionally resource-intensive. Evolutionary and reinforcement learning approaches routinely require thousands-to-millions of candidate evaluations or billions of generated tokens to yield a single verified breakthrough [2, 25], presenting a steep barrier to entry.

Domain Specificity vs. Generality: Most current architectures consist of domain-specific pipelines focused on narrowly defined mathematical areas (e.g., specific graph properties or matrix field operations). Transferring these frameworks across domain boundaries typically requires significant manual re-engineering of reward functions, environment dynamics, or symbolic representations.

The Verification Bottleneck: While neural models can sample candidate conjectures or proofs at a massive scale, formal proof assistants (such as Lean or Isabelle) are inherently slower, single-threaded execution environments. Consequently, the computational bottleneck has shifted from *generating* hypotheses to *verifying* them.

The Quality-vs-Quantity Dilemma: Automated frameworks excel at mass-producing syntactically valid mathematical statements, but the vast majority of these outputs are mathematically trivial, redundant, or tautological. Separating mathematically profound, "aesthetic" insights from mathematically uninteresting truths remains a persistent challenge [24, 36].

Opaque Proof Interpretability: Even when an agent successfully uncovers a novel construction or a formal proof, the resulting artifacts, whether machine-optimized code blocks or highly dense formal proof scripts, are frequently unreadable to human mathematicians, complicating the extraction of broader theoretical intuition.

9.3 Emerging Trends

As the field matures, the research community is coalescing around several promising paradigms designed to alleviate these core limitations:

First, there is a distinct shift toward **autonomous curriculum generation** via self-play frameworks. By decoupling systems into dual conjecturer and prover roles, models like STP [25] and LEGO-Prover [26] allow the system to organically scale problem difficulty in tandem with agent capability, elegantly circumventing the chronic data scarcity that plagues formal mathematics.

Second, the domain is benefiting from a robust **open-source democratization push**. Newly released repositories, such as ShinkaEvolve [19] for sample-efficient code evolution, CoEvo [20] for continuous symbolic tracking, and LeanConjecturer [34], are lowering the computational overhead and providing standardized codebases, encouraging modular community contributions.

Finally, the latest generation of discovery engines is actively **fusing test-time reasoning with evolutionary search**. Frameworks like CPro1 [42] demonstrate that prompting language models to execute deep chain-of-thought or internal reasoning protocols prior to generation drastically increases the hit rate when tackling open mathematical instances. Blending this slow, deliberate inference-time compute with structured evolutionary mutation loops represents the current vanguard of machine-assisted mathematical discovery.

10 Technical Recommendations for Practitioners

Based on our synthesis, we provide actionable recommendations for building mathematical discovery systems. Appendix A summarizes guidance on when practitioners should use specific approaches, depending on the mathematical domain.

Architecture Selection: Practitioners should match the system architecture to the type of discovery. They should use evolutionary methods such as FunSearch and AlphaEvolve for open-ended exploration; self-play and expert-iteration approaches like STP and LEGO-Prover for theorem generation; neuro-symbolic hybrids such as Lemmanaid for lemma discovery; symbolic mutation combined with reinforcement learning for counterexample search; and LLM-based methods combined with symbolic regression, such as LLM-SR, for equation discovery.

Verification Strategy: Systems should employ Lean 4 to meet modern formalization needs. They should integrate verification into the generation loop, provide fine-grained feedback instead of binary pass-fail signals, and incorporate problem-specific evaluators when efficiency is required.

Training Data: Practitioners should rely on self-play to generate conjectures for training. They should apply symbolic mutations to increase diversity, leverage existing formalized mathematics including Mathlib and AFP, and use state-space exploration approaches such as LeanNavigator.

Sample Efficiency: Systems should implement novelty detection, as demonstrated in ShinkaEvolve. They should adopt bandit-based model-selection methods for multi-LLM setups, balance exploration and exploitation during parent selection, and cache and reuse intermediate results to reduce redundant computation.

Evaluation: Practitioners should measure discovery rather than only problem-solving performance, following the ATG benchmark approach. They should prevent memorization, consistent with LLM-SRBench design principles, evaluate the usefulness of generated theorems for downstream tasks, and report both quantity-based and quality-based metrics.

Formal theorem proving and conjecturing (Isabelle/Lean): Practitioners should prefer Lemmanaid, STP, LeanConjecture, LEGO-Prover, and the

in-context Lean method when the goal is to produce stronger proofs and conjectures inside proof assistants. They should use Lemmanaid or LeanConjecturer for conjecture-data generation, STP for self-improving provers, and LEGO-Prover for expanding long-term theorem libraries.

Formal proof ecosystems (Lean, Metamath, PVS): For data generation and benchmarking, practitioners should use ATG and LeanComb with ATG4CI to create or evaluate theorem libraries, and they should adopt LeanNavigator when generating large-scale Lean training corpora. For pipeline design, they should use Lean-FIRe and math-PVS to integrate conjecturing and autoformalization and to connect existing mathematical literature with proof-assistant environments.

Scientific equation discovery and physical systems: Practitioners should use LLM4ED and related equation-discovery frameworks when they can score equations using data. They should combine these frameworks with LLM-SRBench to support fair evaluation.

Conjecture mining and pattern discovery (informal or semi-formal): Practitioners should use the Ramanujan Machine for discovering numeric constants, apply Mining Math Conjectures for algebraic and group-theoretic settings, and adopt Davies *et al.*-style workflows when human intuition and interpretative reasoning play a central role.

Combinatorial design and structure search: For discrete objects, practitioners should use RL-style methods such as Reinforced Generation of Combinatorial Structures or reasoning-based approaches like CPro1. RL approaches are most appropriate when strong scoring functions are available and black-box optimization is desired, whereas CPro1 is suitable when interpretable chain-of-thought design is important.

Frameworks and high-level practice: For practical deployment, practitioners should adopt Generative Modeling for Mathematical Discovery when mathematicians need a plug-and-play FunSearch-based workflow without requiring machine-learning expertise. For broader research-practice guidance, they should combine Carbone's conceptual framework with the human-intuition framework of Davies *et al.* to design human-AI collaborative workflows tailored to their own mathematical domains.

11 Discussion

This section contextualizes the synthesized patterns and empirical breakthroughs of machine-assisted mathematics within a broader theoretical and practical framework. We begin by explicitly addressing the core research questions that structured this systematic review, mapping current technological capacities to their verified mathematical outcomes. Beyond these targeted inquiries, we inspect the bidirectional implications of this advancement, examining how autonomous agents alter the workflow of working mathematicians while simultaneously imposing new design constraints on artificial intelligence practitioners. Finally, we candidly address the systemic limitations inherent to this review and outline the foundational open challenges that remain unresolved at the frontier of artificial mathematical intelligence.

11.1 Answers to Research Questions

We synthesize the comprehensive findings of this systematic review by directly addressing the core research questions posed in Section 1.

RQ1: Technical Architectures and Methodologies

Our analysis isolates seven distinct methodological paradigms currently driving machine-assisted mathematical discovery:

Evolutionary Search: Using LLMs as mutation and crossover operators within open-ended population pools [2, 17, 19, 20].
Reinforcement Learning: Framing discovery as sequential decision-making over discrete game boards or combinatorial states [3, 23].
Self-Play and Expert Iteration: Co-evolving asymmetric conjecturer and prover roles to dynamically scale problem difficulty [25, 26].
Neuro-Symbolic Hybrids: Blending open-ended neural generation with deterministic symbolic execution and template instantiators [29, 30].
In-Context Learning Pipelines: Relying on real-time prompt-context accumulation during inference without modifying core model weights [33, 34].
Specialized Pretraining Objectives: Tailoring self-supervised learning directly to the structural topologies of mathematical syntax trees [35].
Equation Discovery: Grounding parametric LLM priors in strict empirical data regression environments [36].

Key Finding: Frameworks that pair evolutionary program-space search with inline formal verification currently represent the most effective engines for discovering novel, human-interpretable mathematical constructions.

RQ2: Formal Verification Systems

Rigorous, machine-checked validation forms the backbone of autonomous discovery. Lean (specifically Lean 4) has emerged as the dominant interactive theorem prover, driving 28.1% (nine out of 32) of the surveyed literature. The remaining formal ecosystem is comprised of Isabelle/HOL and its Archive of Formal Proofs (two studies), Metamath (one study), and PVS (one study). Additionally, five highly impactful studies bypass generalized provers entirely, utilizing specialized, problem-specific programmatic evaluators to prioritize computational throughput and speed.

Key Finding: The operational paradigm has shifted decisively from post-hoc verification to tight, in-the-loop integration, where execution errors directly shape the generative trajectory.

RQ3: Datasets and Benchmarks

The domain relies on a mixture of massive formal repositories and targeted evaluation suites. Core foundational datasets include Mathlib4 (exceeding one million

lines of verified Lean code), the auto-generated LeanNavigator (4.7 million theorems containing one billion tokens), and LeanComb-Enhanced (260,000 specialized combinatorial identities). Downstream evaluation is anchored by miniF2F (488 problems across olympiad scales), LLM-SRBench (239 cross-domain problems featuring anti-memorization syntax transforms), and ConjectureBench.

Key Finding: Advanced self-play and closed-loop synthetic data generation have proven highly capable of mitigating the chronic scarcity of human-formalized mathematical data.

RQ4: Success Factors and Limitations

Success is consistently dictated by the transition from single-shot text generation to iterative, stateful loops. Top-performing systems exploit deep domain-specific structures, maintain evolving code databases, and generate internal testing curricula via autonomous feedback. Conversely, these systems remain bounded by immense computational costs (often requiring millions of iterations), extreme narrowness of scope, verification bottlenecks caused by single-threaded execution environments, and an unfavorable quality-to-quantity ratio that yields countless correct but trivial tautologies.

Key Finding: Success is driven by iterative, stateful self-improvement rather than single-shot generation, but current approaches remain limited by computational costs, narrow specialization, and verification bottlenecks.

RQ5: Mathematical Domains Addressed

By publication volume, the mathematical fields addressed scale as follows: **combinatorics and graph theory** leads significantly (twelve studies), followed by **algebra and number theory** (eight studies), **algorithm design** (four studies), **geometry** (four studies), **analysis and PDEs** (three studies), **complexity theory** (two studies), and **topology** (two studies).

Key Finding: Combinatorics has received disproportionate academic attention because its search spaces are inherently discrete, bounded, and easily mapped onto explicit, programmatic testing harnesses.

11.2 Implications for Mathematical Research

The findings of this review show that learning-driven mathematical discovery has moved from a largely theoretical idea to a practial research approach. Approaches such as FunSearch and AlphaEvolve have been used to improve algorithms and mathematical lower bounds that had not changes for more than 50 years, demonstrating their potential to contribute to mathematical research.

Human-AI Epistemic Complementarity: Rather than rendering human mathematicians obsolete, these technologies point toward a deeply collaborative

future. As demonstrated by the landmark knot-theory work of Davies *et al.* [16], machine-learning systems excel at unearthing hidden statistical patterns and counterexamples across massive dimensions, which human experts can then synthesize into overarching theoretical frameworks.

Democratization and Research Accessibility: The proliferation of open-source, lightweight implementations of evolutionary search loops significantly lowers the barrier to entry. Working mathematicians can now use specialized discovery engines locally, using automated systems as specialized "ideation sandboxes."

Transition to Bidirectional Exploration: Automated mathematics is no longer a one-way street of proving human-written statements. Systems like TongGeometry [24] demonstrate that neural architectures can operate bidirectionally, proposing aesthetically valuable, competitive-grade conjectures and problems alongside formal proofs.

11.3 Implications for LLM Development

For AI practitioners and computer scientists engineering the next generation of foundation models, this literature highlights four critical design directives:

The Inseparability of Execution and Generation: For safety-critical and highly rigorous domains like mathematics, standalone generation is insufficient. Hallucinations are fundamentally catastrophic in formal logic; therefore, LLM development must prioritize architectures natively built to interface with external execution environments and compilers.

The Limitations of Autoregressive Fine-Tuning: Standard next-token prediction over linear text sequences fundamentally underrepresents the nested, nonlinear dependencies of formal proofs. Pretraining objectives must evolve toward structure-aware strategies, such as Skip-Tree objectives [35], which explicitly force the model to learn the semantic structure of ASTs.

Bootstrapping via Synthetic Playmills: Data scarcity in specialized fields cannot be solved by web-scraping alone. Building dual-agent architectures that autonomously balance difficulty and verification capacity provides a sustainable blueprint for generating pristine, high-fidelity synthetic pretraining corpora.

The Superiority of Programmatic Abstraction: Searching for a specific mathematical solution directly is structurally fragile. Forcing models to operate in the space of functional, parameter-driven *programs* that subsequently synthesize the solution introduces a layer of abstraction that yields dramatically higher generalizability and human interpretability.

11.4 Limitations of This Review

We explicitly acknowledge several methodology-driven limitations inherent to this review:

Rapid Temporal Obsolescence: The intersection of LLMs and formal mathematics is evolving at an unprecedented pace. Given the velocity of weekly

preprint distributions, breakthroughs published immediately prior to print may be underrepresented.

Systemic Publication Bias: The literature exhibits a heavy positive-outcome bias. Systems that failed to find novel bounds or resolve known conjectures are rarely documented, obscuring valuable insights regarding what architectural configurations do *not* work.

Evaluation Heterogeneity: There is a distinct lack of standardized benchmarking across different studies. The reliance on disparate metrics, varied hardware allocations, and highly custom problem domains prevents a rigorous, quantitative head-to-head performance comparison.

Single-Reviewer Screening: While the literature mapping was conducted via strict, systematic protocols, the screening and extraction phases were executed by a single primary investigator rather than multiple independent human coders, introducing a minor risk of subjective selection bias.

11.5 Open Challenges

As the field moves forward, five foundational challenges remain entirely unresolved.

The Generality Gap: Current systems are heavily siloed. An agent engineered to discover matrix multiplication algorithms cannot choose to pivot and construct a topological counterexample without substantial human re-engineering. True cross-domain generalization remains elusive.

The Creative Ceiling: Automated discoveries are overwhelmingly incremental. They tighten existing numerical bounds, optimize established algorithms, or find hidden edge cases. Transformative conceptual leaps, such as inventing entirely new branches of mathematics or proposing paradigm-shifting definitions, remain uniquely human.

Formal Script Legibility: While an interactive theorem prover can certify that a machine-generated proof script is 100% correct, these files are frequently comprised of dense, unreadable, and non-intuitive token structures. Transforming valid formal proofs into semantic, human-readable explanations is an urgent, open problem.

Asymmetric Compute Requirements: The environmental and financial cost of running massive evolutionary search loops or self-play regimes restricts this frontier to well-funded industrial and academic institutions, threatening to gatekeep AI-driven mathematical advancement.

Objective Metric for "Interestingness": Computers can generate billions of valid tautologies, but they struggle to evaluate mathematical beauty or theoretical utility. Developing mathematical reward models capable of reliably quantifying the qualitative elegance and deeper importance of a discovery is perhaps the field's most critical hurdle.

12 Future Directions

Based on our synthesis, we identify the following promising research directions.

Integration with reasoning models: The emergence of reasoning-capable LLMs such as o1 and DeepSeek-R1 presents opportunities for chain-of-thought reasoning before conjecture generation, self-reflection on generated mathematical objects, and multi-step planning for proof construction.

Multi-modal mathematical discovery: Geometric discovery, as demonstrated by TongGeometry, shows the value of visual reasoning, suggesting opportunities for integration of diagram understanding, visual pattern recognition for conjecture formation, and multi-modal verification systems.

Collaborative human-AI discovery: The Davies *et al.* model suggests to develop interactive systems for mathematician-AI collaboration, explainable discovery processes, and tools that enhance rather than to replace mathematical intuition.

Scaling and efficiency: Addressing computational limitations requires more sample-efficient evolutionary algorithms, parallel verification systems, and incremental learning to avoid retraining.

13 Conclusion

This systematic review examined 32 studies on mathematical discovery using LLMs published through December 2025, following the PRISMA 2020 guidelines. We identified seven distinct technical paradigms and synthesized findings across formal verification systems, datasets, and evaluation approaches.

Our key findings follow. Evolutionary search combined with formal verification has produced the most significant mathematical discoveries, including improvements to algorithms unchanged for over 50 years. Self-play and expert iteration effectively address the data scarcity challenge in formal mathematics by enabling systems to generate their own training data. Lean has emerged as the dominant verification system, used by over 27% of included studies, benefiting from its active development and extensive library. Neuro-symbolic hybrid approaches offer promising avenues for combining the generative capabilities of LLMs with the rigor of symbolic methods. Open-source implementations are democratizing access to mathematical discovery tools, enabling broader participation in this research area.

The field of LLM-based mathematical discovery has demonstrated remarkable progress in a short time, with results published in top venues. As systems become more capable and accessible, we anticipate an increase in the integration of AI tools into mathematical research workflows, enhancing human creativity and intuition with computational power and pattern recognition. The ultimate goal is to create AI systems that can independently advance mathematical knowledge, which remains a challenge but increasingly within reach.

Data Availability Statement

The complete dataset of included studies, screening decisions, and extracted data supporting this systematic review is available in the supplementary materials. The original literature search results were obtained from publicly accessible databases including

Semantic Scholar, arXiv, and OpenReview. All code repositories referenced in included studies are cited with their respective URLs, where available.

Funding

This work was supported by the National Science Foundation under Award No. 2432704.

Conflict of Interest

The authors declare no conflicts of interest.

A Comparative analysis of included approaches

This table presents a comparative analysis of the included approaches, offering detailed insights into their primary domains and use cases. It highlights the core techniques employed, outlines the methodological novelties, and provides guidance for researchers seeking to adopt these methods. Additionally, the table emphasizes the key innovations introduced by each study. Overall, this consolidated information is intended to serve as a resource for both mathematicians and computer science practitioners who are interested in leveraging LLMs for mathematical conjecturing.

Lemmanaid [30]

Primary domain & use case: Interactive formalization in Isabelle, lemma discovery to support existing proof developments.

Core technique & and utility: Neuro-symbolic lemma conjecturing: an LLM proposes lemma templates and a symbolic engine instantiates and checks them, giving both breath (neural) and rigor (symbolic).

Methodological novelty: First bottom-up neuro-symbolic lemma-template pipeline in Isabelle. Focuses on analogical lemma families rather than full statements.

User guidance: Use when extending a formal library and needing many useful mid-level lemmas that fit a given theory context in Isabelle-HOL.

Innovation: Template-based conjecturing tightly coupled to a proof assistant, balancing neural creativity with symbolic correctness.

Continued on next page

STP [25]

Primary domain & use case: Automated theorem proving in Lean/Isabelle with self-improving conjecturer/prover loop.

Core technique & and utility: Self-play between conjecturer and prover: the conjecturer generates statements "just hard enough," and the prover trains via expert iteration, yielding a curriculum of increasingly challenging theorems.

Methodological novelty: Unifies iterative conjecturing and proving into one training loop. Turns theorem proving into a self-play game with no fixed dataset.

User guidance: Choose STP for large-scale ATP benchmarks where an LLM prover is needed to keep improving without hand-curated conjecture sets.

Innovation: Tight conjecture-prover co-evolution that doubles prior LeanWorkbook success rates.

RL matrix multiplication [3]

Primary domain & use case: Algorithm discovery for structured algebraic problems, especially matrix multiplication.

Core technique & and utility: Reinforcement learning over algorithmic search spaces (tensor decompositions) guided by a performance score (number of multiplications).

Methodological novelty: One of the first RL systems to discover new, human-competitive matrix multiplication schemes beyond Strassen-type constructions.

User guidance: Use for tightly-specified algorithmic design problems with clear discrete structures and evaluation metrics (e.g., tensor/graph decompositions).

Innovation: Deep cross-entropy style RL tailored to symbolic tensor factorization for faster matrix multiplication.

Mathematical exploration at scale [18]

Primary domain & use case: Broad mathematical exploration (e.g., combinatorics, geometry, or analysis) using AlphaEvolve-style search.

Core technique & and utility: LLM-guided evolutionary program search over constructive objects, with large-scale experimental campaigns across dozens of problems.

Methodological novelty: Systematic, multi-problem validation of LLM-driven constructive search as a general methodology for "mathematics at scale."

User guidance: Use when needing a framework and empirical blueprint for running many AlphaEvolve-like experiments across diverse problems.

Innovation: Demonstrates that a single evolutionary coding paradigm can push bounds on many unrelated open problems.

Continued on next page

CoEvo [20]

Primary domain & use case: Open-ended symbolic discovery (e.g., equations, code, or expressions) across scientific/engineering tasks.

Core technique & and utility: Evolutionary search with LLMs plus a dynamic knowledge library and multi-modal representations (e.g., text, math, or code), supporting continual, lifelong discovery.

Methodological novelty: First to explicitly frame symbolic discovery as continual co-evolution of solutions and a shared knowledge base.

User guidance: Use CoEvo when problems evolve over time and a reusable knowledge memory (e.g., ongoing modeling or design pipelines) is needed.

Innovation: Coupling of evolutionary LLM search with an evolving, centralized knowledge library across representation spaces.

Conjecturing with LLMs [29]

Primary domain & use case: Early, small-scale exploration of using LLMs for informal mathematical conjecturing.

Core technique & and utility: Prompt-based use of general LLMs to generate conjectures and analyze their plausibility, often outside fully formal settings.

Methodological novelty: Among the first systematic case studies of LLMs as conjecturing partners in mathematics

User guidance: Use as a conceptual baseline or lightweight approach when informal ideas or pilot studies without full formalization are needed.

Innovation: Positions generic LLMs as creative, dialog-based conjecture generators for human mathematicians.

In-context Lean theorem discovery [33]

Primary domain & use case: Discovery of new theorems within Lean via in-context learning from existing proofs.

Core technique & and utility: Treats existing Lean proofs as in-context demonstrations so the LLM can propose new theorems and proofs that mirror library patterns, enabling discovery beyond training data.

Methodological novelty: Leverages in-context proof learning (rather than fine-tuning) specifically for new theorem statements in a formal system.

User guidance: Use with a rich Lean corpus where discovery tightly anchored to its style and idioms without complex training pipelines is desired.

Innovation: In-context proof pattern mining for new theorem discovery in a concrete proof assistant.

Continued on next page

ShinkaEvolve [19]

Primary domain & use case: Open-ended program/agent evolution, including scientific code and reasoning scaffolds.

Core technique & and utility: Archive-based evolutionary program search with novelty-aware rejection sampling and bandit-based LLM ensemble selection, emphasizing sample efficiency.

Methodological novelty: Shows open-ended program evolution that is orders of magnitude more sample-efficient than prior AlphaEvolve-style systems.

User guidance: Use when evaluations are expensive (e.g., complex simulators or costly scientific objectives) and efficient exploration of program spaces is needed.

Innovation: Integration of novelty rejection sampling and adaptive ensemble selection into LLM-driven evolutionary search.

AlphaEvolve [17]

Primary domain & use case: General coding agent for scientific/algorithmic discovery (e.g., mathematics, algorithms, or systems).

Core technique & and utility: LLM-guided evolutionary search over code "programs" with automatic evaluation metrics, tuned for large parallel compute.

Methodological novelty: Demonstrates that such agents can surpass human records on multiple open problems and infrastructure tasks, not just isolated benchmarks.

User guidance: Use AlphaEvolve-style setups with a reliable numerical evaluator and large-scale compute, and want strong solutions rather than interpretability.

Innovation: Highly scalable, domain-agnostic evolutionary coding agent that treats discoveries as high-scoring programs.

LeanConjecturer [34]

Primary domain & use case: Automatic generation of Lean-4 conjectures to augment formal theorem-proving datasets.

Core technique & and utility: Hybrid pipeline: rule-based context extraction from Mathlib plus LLM-based theorem-statement generation and automated filtering of trivialities.

Methodological novelty: Large-scale Lean conjecture generation (12k+ conjectures, with thousands non-trivial) specifically targeting data scarcity for formal provers.

User guidance: Use when needing many diverse and non-trivial Lean statements for training or evaluating formal theorem provers.

Innovation: Pipeline tackling the "triviality trap" in formal conjecture generation via domain-aware context and filters.

Continued on next page

Skip-tree training [35]

Primary domain & use case: Pretraining neural models for symbolic and mathematical reasoning.

Core technique & and utility: Self-supervised skip-tree objective over proof trees/theory graphs to learn structural reasoning patterns.

Methodological novelty: Introduces a pretraining task tailored to tree-structured mathematical objects rather than generic text.

User guidance: Use when building base models intended to serve as theorem provers or proof assistants.

Innovation: Task design that aligns representation learning with proof-tree structure.

FunSearch [2]

Primary domain & use case: Program-search-based discovery for extremal combinatorics and algorithms (e.g., cap sets or bin packing).

Core technique & and utility: Island-based evolutionary search in function space using LLM-generated programs and automatic scoring functions.

Methodological novelty: Shows LLM program search can yield genuinely new mathematical results and scalable heuristics, not just proofs.

User guidance: Use FunSearch-style methods with a crisp scoring function and care about interpretable heuristic code (e.g., combinatorial constructions or greedy algorithms).

Innovation: Function-space evolutionary search producing interpretable, human-inspectable code with provably new constructions.

LLM-SRBench [36]

Primary domain & use case: Benchmarking equation and symbolic regression discovery by LLMs in scientific domains.

Core technique & and utility: Standardized benchmark with curated tasks and metrics for scientific equation discovery.

Methodological novelty: First broad benchmark targeted at LLM-based equation discovery, enabling fair comparison of methods.

User guidance: Use when evaluating new symbolic discovery methods on real-world equation-discovery tasks.

Innovation: Provides a "yardstick" for LLM-based symbolic regression in science.

Continued on next page

Combinatorics via NNs [23]

Primary domain & use case: Extremal combinatorics constructions and counterexamples.

Core technique & and utility: Deep cross-entropy reinforcement learning to search over combinatorial structures guided by problem-specific score functions.

Methodological novelty: Early, influential demonstration that deep RL can find explicit extremal constructions and refute conjectures.

User guidance: Use for discrete combinatorial design problems with reasonably fast evaluators and small-to-medium-sized search spaces.

Innovation: Application of deep RL as a generic engine for combinatorial construction search.

LEGO-Prover [26]

Primary domain & use case: Neural theorem proving with reusable lemma libraries.

Core technique & and utility: LLM-driven prover augmented by a growing library of "skills" (lemmas) that are created, reused, and evolved during proof search.

Methodological novelty: Makes lemma discovery endogenous to proving, leading to a self-expanding skill library that improves success rates.

User guidance: Use for long-term theorem-proving systems whose capabilities grow as they solve more problems.

Innovation: Treats verified lemmas as modular, composable skills that accumulate over time.

Olympiad geometry via guided search [24]

Primary domain & use case: Solving and posing Olympiad-level geometry problems.

Core technique & and utility: Guided tree search over geometric constructions and theorems, coupled with symbolic geometry reasoning and problem-generation routines.

Methodological novelty: Joint framework that both proposes new geometry problems and solves them at competition difficulty.

User guidance: Use for highly structured, diagram-based domains where symbolic geometry engines and search can be tightly integrated (e.g., contest-style geometry).

Innovation: Coupled problem-posing and solving pipeline in a symbolic-geometric setting.

Continued on next page

Combinatorial Identities Benchmark via ATG4CI [38]

Primary domain & use case: Lean theorem proving on combinatorial identities and data generation for ATP.

Core technique & and utility: ATG4CI: LLM-guided tactic prediction plus RL tree search to auto-generate new combinatorial-identity theorems and proofs, forming a large LeanComb-Enhanced dataset.

Methodological novelty: First dedicated Lean benchmark and automated theorem generator for combinatorial identities, producing 260k+ fully proved theorems.

User guidance: Use when studying ATP in combinatorics or needing large, domain-specific Lean training data for proof models.

Innovation: Tight coupling of tactic-level RL and LLM suggestions to expand a formal benchmark in a focused domain.

Guiding human intuition with AI [16]

Primary domain & use case: Human-in-the-loop pure math (e.g., topology or representation theory).

Core technique & and utility: Supervised ML discovers patterns between mathematical objects. Attribution and visualization guide human conjecture formation and proof strategy.

Methodological novelty: A general workflow where ML acts as a "test bed for intuition" that led to new results in knot theory and the combinatorial invariance conjecture.

User guidance: Use when ML is a tool for insight (e.g., feature importance or pattern finding) rather than autonomous theorem proving.

Innovation: Conceptual framework for AI as a collaborator that identifies patterns which humans then turn into rigorous conjectures and proofs.

Advancing mathematics research with generative AI [43]

Primary domain & use case: Position/vision for generative-AI-assisted mathematical research.

Core technique & and utility: Conceptual analysis of how generative models may support conjecturing, proof exploration, and exposition across subfields.

Methodological novelty: Synthesizes emerging systems into guidance for mathematicians rather than proposing a specific algorithm.

User guidance: Use for high-level strategy and ethical/practical framing when planning AI integration into a research program.

Innovation: Maps concrete AI capabilities to stages of mathematical practice (e.g., intuition, search, verification or exposition).

Continued on next page

ATG benchmark [41]

Primary domain & use case: Evaluating automated theorem generation using the Metamath library.

Core technique & and utility: Splits Metamath into axioms/library/problems and measures whether generated theorems improve downstream proving.

Methodological novelty: First benchmark where success is measured by generating new, reusable theorems that help prove harder targets.

User guidance: Use to evaluate a conjecture or theorem generator, not just a prover, in a clean ATP setting.

Innovation: Treats theorem generation as a benchmarked task with impact measured on downstream ATP performance.

Massive empirical theorems via forward reasoning [44]

Primary domain & use case: Automated theorem finding and knowledge expansion using non-LLM logic-based methods.

Core technique & and utility: Forward reasoning on strong relevant logics generates many "empirical" theorems, aiming at scalable theorem discovery.

Methodological novelty: Proposes empirical theorem generation as a data source when conventional pretraining data is exhausted.

User guidance: Use as a logic-driven complement to LLMs when large pools of plausible theorems for training or exploration are needed.

Innovation: Reframes theorem generation as large-scale, logic-based data synthesis for AI systems.

Conjecturing: an overlook step [40]

Primary domain & use case: Autoformalization and conjecture generation in Lean.

Core technique & and utility: ConjectureBench dataset with metrics ConJudge and equiv_rfl along with the Lean-FIRe inference-time framework to improve conjecturing inside autoformalization pipelines.

Methodological novelty: Separates conjecturing from proof and formalization, showing current LLM performance is overestimated when the conjecture is given

User guidance: Use to rigorously measure and improve the conjecture step in formal reasoning systems.

Innovation: Recasting conjecturing as a first-class, benchmarked task with metrics and the first end-to-end autoformalization of Putnam-level problems.

Continued on next page

Ramanujan Machine [15]

Primary domain & use case: Conjectures about fundamental constants (e.g., π, e, and ζ).

Core technique & and utility: Numeric search (e.g., meet-in-the-middle or specialized gradient methods) that matches numerical values to symbolic continued-fraction forms.

Methodological novelty: Systematic, fully automated conjecturing pipeline for constants, producing both known and new formulas.

User guidance: Use when the domain is numerical constants or series with high-precision evaluations rather than formal proofs.

Innovation: Treats conjecturing as numerical pattern detection that bypasses explicit structural understanding.

LeanNavigator [39]

Primary domain & use case: Large-scale synthetic Lean theorem generation.

Core technique & and utility: Agentic framework that navigates Lean libraries to generate and verify millions of theorems for training provers.

Methodological novelty: Focuses on scale with millions of verified Lean theorems as training data, beyond earlier smaller datasets.

User guidance: Use for pretraining or augmenting large Lean-based theorem provers needing massive, library-compatible data.

Innovation: Industrial-scale generation of new, machine-checked formal theorems.

Generative Modeling for Mathematical Discovery [45]

Primary domain & use case: Practical FunSearch-style discovery for working mathematicians.

Core technique & and utility: Accessible implementation of the LLM-driven genetic programming FunSearch where users only modify a small Python stub and choose an LLM provider.

Methodological novelty: Emphasizes usability and benchmarking of FunSearch on several new problems, without requiring HPC or ML expertise.

User guidance: Use for function-space program search with combinatorial and number-theoretic problems with modest resources.

Innovation: Packaging an LLM-genetic algorithm pipeline as a practical, plug-and-play tool for mathematicians.

Continued on next page

Language Modeling for Formal Mathematics [46]

Primary domain & use case: General language modeling over formal corpora (e.g., HOL Light or Mizar).

Core technique & and utility: Sequence modeling of formal statements and proofs as a foundation for later theorem-proving and conjecturing models.

Methodological novelty: Early demonstration that large neural LMs can model formal mathematics at scale.

User guidance: Use as a foundational dataset or model reference when designing new formal-math LMs.

Innovation: Positions formal corpora as a natural domain for large language modeling, and seeding later ATP work.

LLMs for automatic equation discovery [47]

Primary domain & use case: Data-driven discovery of ordinary and partial differential equations governing dynamics.

Core technique & and utility: Prompt-based framework LLM4ED, where LLMs generate candidate equations and self-improve via alternating self-improvement and evolutionary search, using data-based scores.

Methodological novelty: One of the first plug-and-play LLM frameworks for equation discovery that matches or exceeds classic symbolic regression methods.

User guidance: Use for physical or scientific systems where it is possible to simulate or measure trajectories. Moreover, use in cases where the governing equations are desired with minimal coding.

Innovation: Natural-language-driven equation search which combine LLM self-improvement and evolutionary operators.

---

math-PVS [48]

Primary domain & use case: Mapping informal mathematics in papers to PVS formalizations.

Core technique & and utility: LLM-based framework that parses natural-language mathematics and aligns it with PVS theories and objects.

Methodological novelty: Bridges literature and a specific proof assistant (PVS) rather than focusing purely on theorem proving.

User guidance: Use to connect published results to a mechanized repository in PVS (e.g., for safety-critical verification).

Innovation: Treats the literature to proof assistant mapping as a structured LLM task at scale.

Continued on next page

Mining Math Conjectures from LLMs [49]

Primary domain & use case: Conjecture mining in group theory using solubilizer.

Core technique & and utility: Two-stage pipeline where LLMs generate conjectures, and then LLMs with GAP search for counterexamples to prune obviously false ones.

Methodological novelty: Shows that generic LLMs can yield original, testable conjectures when combined with automated counterexample search, without a full proof assistant.

User guidance: Use for exploratory conjecture mining in algebraic domains with accessible computer-algebra systems.

Innovation: Guess-and-check conjecturing loop that leverages LLM creativity and automated falsification.

Reinforced Generation of Combinatorial Structures [50]

Primary domain & use case: Design of combinatorial objects (e.g., graphs or codes).

Core technique & and utility: Reinforcement-learning or policy-gradient generation of discrete structures guided by problem-specific scores.

Methodological novelty: Extends RL-style search to new combinatorial design settings with tailored reward shaping.

User guidance: Use with a clear scoring function on discrete objects, but not necessarily in a formal proof setting.

Innovation: General RL framework for synthesizing combinatorial structures beyond specific extremal problems.

Self-reflecting LLMs: Hegelian Dialectic [51]

Primary domain & use case: Meta-reasoning and self-critique in LLM problem solving including mathematics.

Core technique & and utility: Iterative thesis-antithesis-synthesis prompting where the model attacks and refines its own reasoning chains.

Methodological novelty: Introduces a philosophical self-reflection pattern that improves robustness and exploration in reasoning tasks.

User guidance: Use as a prompting/control layer on top of mathematical-reasoning models for systematic self-critique without new training.

Innovation: Hegelian-style dialectic loop applied to LLM reasoning traces.

Continued on next page

CPro1 [42]

Primary domain & use case: Combinatorial design using chain-of-thought reasoning models.

Core technique & and utility: Uses reasoning-enhanced LLMs to iteratively refine candidate designs, often with external evaluation.

Methodological novelty: Moves from pure RL or heuristic search to explicit chain-of-thought-style reasoning for design choices.

User guidance: Use with reliable LLM-internal reasoning (not just scores) to guide combinatorial design decisions.

Innovation: Treats combinatorial design as an interactive dialogue between a reasoning LLM and an evaluator, rather than black-box search.

B Complete List of Included Studies

Table 1 presents the complete list of 32 studies included in this systematic review, ordered alphabetically by title.

Table 1: Complete List of 32 Included Studies

#	Title	Authors	Year	Venue
[38]	A combinatorial identities benchmark for theorem proving via automated theorem generation	Xiong *et al.*	2025	arXiv
[16]	Advancing mathematics by guiding human intuition with AI	Davies *et al.*	2021	Nature
[43]	Advancing mathematics research with generative AI	Carbone	2025	arXiv
[17]	AlphaEvolve: A coding agent for scientific and algorithmic discovery	Novikov *et al.*	2025	arXiv
[41]	ATG: Benchmarking automated theorem generation for generative language models	Lin *et al.*	2024	NAACL
[44]	Automated generation of massive reasonable empirical theorems by forward reasoning based on strong relevant logics	Cheng	2024	arXiv
[20]	CoEvo: Continual evolution of symbolic solutions using large language models	Guo *et al.*	2024	arXiv
[40]	Conjecturing: An overlooked step in formal mathematical reasoning	Sivakumar *et al.*	2025	arXiv
[23]	Constructions in combinatorics via neural networks	Wagner	2021	arXiv

Continued on next page

Table 1 – continued from previous page

#	Title	Authors	Year	Venue
[3]	Discovering faster matrix multiplication algorithms with reinforced learning	Fawzi *et al.*	2022	Nature
[33]	Discovering new theorems via LLMs with in-context proof learning in Lean	Kasaura *et al.*	2025	arXiv
[29]	Exploring mathematical conjecturing with large language models	Johansson and Smallbone	2023	NeSy
[15]	Generating conjectures on fundamental constants with the Ramanujan Machine	Raayoni *et al.*	2021	Nature
[39]	Generating millions of Lean theorems with proofs by exploring state transition graphs	Yin and Gao	2025	arXiv
[45]	Generative modeling for mathematical discovery	Ellenberg *et al.*	2025	arXiv
[46]	Language modeling for formal mathematics	Rabe *et al.*	2020	arXiv
[47]	Large language models for automatic equation discovery of nonlinear dynamics	Du *et al.*	2024	Phys. Fluids
[34]	LeanConjecturer: Automatic generation of mathematical conjectures for theorem proving	Onda *et al.*	2025	arXiv
[26]	LEGO-Prover: Neural theorem proving with growing libraries	Xin *et al.*	2023	arXiv
[30]	Lemmanaid: Neuro-symbolic lemma conjecturing	Alhessi *et al.*	2025	arXiv
[36]	LLM-SRBench: A new benchmark for scientific equation discovery with large language models	Shojaee *et al.*	2025	ICML
[48]	math-PVS: A large language model framework to map scientific publications to PVS theories	Saidi *et al.*	2023	arXiv
[2]	Mathematical discoveries from program search with large language models	Romera-Paredes *et al.*	2024	Nature
[18]	Mathematical exploration and discovery at scale	Georgiev *et al.*	2025	arXiv
[35]	Mathematical reasoning via self-supervised skip-tree training	Rabe *et al.*	2020	ICLR

Continued on next page

Table 1 – continued from previous page

#	Title	Authors	Year	Venue
[49]	Mining math conjectures from LLMs: A pruning approach	Chuharski *et al.*	2024	NeurIPS WS
[24]	Proposing and solving olympiad geometry with guided tree search	Zhang *et al.*	2024	arXiv
[50]	Reinforced generation of combinatorial structures: Hardness of approximation	Nagda *et al.*	2025	arXiv
[25]	STP: Self-play LLM theorem provers with iterative conjecturing and proving	Dong *et al.*	2025	ICML
[51]	Self-reflecting large language models: A Hegelian dialectical approach	Abdali *et al.*	2025	arXiv
[19]	ShinkaEvolve: Towards open-ended and sample-efficient program evolution	Lange *et al.*	2025	arXiv
[42]	Using reasoning models to generate search heuristics that solve open instances of combinatorial design problems	Rosin	2025	arXiv

C Keyword Frequency Analysis

The most frequently appearing keywords across included studies are Lean and Lean 4 (appearing in twelve studies), conjecture generation (eight studies), theorem proving (seven studies), formal verification (seven studies), evolutionary search (six studies), reinforcement learning (five studies), counterexample (five studies), self-play (four studies), symbolic regression (three studies), and neuro-symbolic approaches (three studies).

D Exclusion Details

Table 2 summarizes the exclusion reasons for a representative sample of excluded studies and does not encompass the full set of excluded studies.

Table 2: Detailed Exclusion Reasons for Representative Studies

Study Title	Code	Explanation
Absolute Zero: Reinforced self-play reasoning with zero data [52]	E1	Focuses on solving problems, not generating new knowledge

Continued on next page

Table 2 – continued from previous page

Study Title	Code	Explanation
Solving olympiad geometry without human demonstrations [53]	E1	Problem solving only
Towards Solving More Challenging IMO Problems [54]	E1	Problem solving only
BioinspiredLLM: Conversational large language model for the mechanics of biological and bio-inspired materials [55]	E2	Non-mathematical domain (biology)
SciAgents: Automating scientific discovery through bioinspired multi-agent intelligent graph reasoning [56]	E2	Non-mathematical domain (materials)
Monte Carlo thought search: Large language model querying for complex scientific reasoning in catalyst design [57]	E2	Non-mathematical domain (chemistry)
AI-Researcher: Autonomous scientific innovation [58]	E2	General scientific discovery
Causal structure learning supervised by large language model [59]	E2	Causal inference, not mathematics
ChatRule: Mining logical rules with Large Language Models for Knowledge Graph Reasoning [60]	E2	Knowledge graphs, not mathematics
Dynamic models of neural population dynamics [61]	E2	Non-mathematical domain (neuroscience)
Recent advances and future directions in literature-based discovery [62]	E2	Non-mathematical domain (biology)
LLM4Laser: Large language models automate the design of lasers [63]	E2	Non-mathematical domain (hardware)
Universal differential equations for glacier ice flow modelling [64]	E2	Non-mathematical domain (geoscience)
Sentiment-aware stock price prediction with transformer and LLM-generated formulaic alpha [65]	E2	Non-mathematical domain (finance)
Math Problem Solving: Enhancing Large Language Models with Semantically Rich Symbolic Variables [66]	E3	Tool calling surveys describe orchestration patterns across math agents.
FunSearch [67]	E4	Duplicate of included study
Agent0: Unleashing self-evolving agents from zero data via tool-integrated reasoning [68]	E4	General agent framework
Hypothesis search: Inductive reasoning with language models[69]	E4	General reasoning, not math-specific

E PRISMA 2020 Checklist

This review was conducted following PRISMA 2020 guidelines. The key checklist items are addressed as follows: the title identifies this as a systematic review, the abstract provides a structured summary, Section 1.1 explains the rationale and motivation, and Section 1.2 states the research questions as objectives. Section 2.1 defines eligibility criteria I1-I5 and E1-E6, Section 2.2 lists information sources and databases, Section 2.2 provides the search strategy and keywords, Section 2.3 describes the two-stage selection process, Section 2.4 lists data extraction variables, and Section 2.5 notes risk of bias limitations. Section 3.1 presents the PRISMA flow diagram for study selection results, Section 3.2 provides tables showing study characteristics. Sections 4-8 present detailed discussion or results, and Section 10 provides interpretation in the discussion.

References

[1] Newell, A., Simon, H.: The logic theory machine–a complex information processing system. IRE Transactions on Information Theory **2**(3), 61–79 (1956) https://doi.org/10.1109/TIT.1956.1056797

[2] Romera-Paredes, B., Barekatain, M., Novikov, A., Balog, M., Kumar, M.P., Dupont, E., Ruiz, F.J., Ellenberg, J.S., Wang, P., Fawzi, O., Kohli, P., Fawzi, A.: Mathematical discoveries from program search with large language models. Nature **625**(7995), 468–475 (2024) https://doi.org/10.1038/s41586-023-06924-6

[3] Fawzi, A., Balog, M., Huang, A., Hubert, T., Romera-Paredes, B., Barekatain, M., Novikov, A., Ruiz, F.J., Schrittwieser, J., Swirszcz, G., Silver, D., Hassabis, D., Kohli, P.: Discovering faster matrix multiplication algorithms with reinforcement learning. Nature **610**(7930), 47–53 (2022) https://doi.org/10.1038/s41586-022-05172-4

[4] Page, M.J., McKenzie, J.E., Bossuyt, P.M., Boutron, I., Hoffmann, T.C., Mulrow, C.D., Shamseer, L., Tetzlaff, J.M., Akl, E.A., Brennan, S.E., Chou, R., Glanville, J., Grimshaw, J.M., Hróbjartsson, A., Lalu, M.M., Li, T., Loder, E.W., Mayo-Wilson, E., McDonald, S., McGuinness, L.A., Stewart, L.A., Thomas, J., Tricco, A.C., Welch, V.A., Whiting, P., Moher, D.: The PRISMA 2020 statement: an updated guideline for reporting systematic reviews. BMJ **372** (2021) https://doi.org/10.1136/bmj.n71

[5] OpenAI, Achiam, J., Adler, S., Agarwal, S., Ahmad, L., Akkaya, I., Aleman, F.L., Almeida, D., Altenschmidt, J., Altman, S., Anadkat, S., Avila, R., Babuschkin, I., Balaji, S., Balcom, V., Baltescu, P., Bao, H., Bavarian, M., Belgum, J., Bello, I., Berdine, J., Bernadett-Shapiro, G., Berner, C., Bogdonoff, L., Boiko, O., Boyd, M., Brakman, A.-L., Brockman, G., Brooks, T., Brundage, M., Button, K., Cai, T., Campbell, R., Cann, A., Carey, B., Carlson, C., Carmichael, R., Chan, B., Chang, C., Chantzis, F., Chen, D., Chen, S., Chen, R., Chen, J., Chen, M., Chess, B., Cho, C., Chu, C., Chung, H.W., Cummings, D., Currier, J., Dai, Y.,

Decareaux, C., Degry, T., Deutsch, N., Deville, D., Dhar, A., Dohan, D., Dowling, S., Dunning, S., Ecoffet, A., Eleti, A., Eloundou, T., Farhi, D., Fedus, L., Felix, N., Fishman, S.P., Forte, J., Fulford, I., Gao, L., Georges, E., Gibson, C., Goel, V., Gogineni, T., Goh, G., Gontijo-Lopes, R., Gordon, J., Grafstein, M., Gray, S., Greene, R., Gross, J., Gu, S.S., Guo, Y., Hallacy, C., Han, J., Harris, J., He, Y., Heaton, M., Heidecke, J., Hesse, C., Hickey, A., Hickey, W., Hoeschele, P., Houghton, B., Hsu, K., Hu, S., Hu, X., Huizinga, J., Jain, S., Jain, S., Jang, J., Jiang, A., Jiang, R., Jin, H., Jin, D., Jomoto, S., Jonn, B., Jun, H., Kaftan, T., Kaiser, Kamali, A., Kanitscheider, I., Keskar, N.S., Khan, T., Kilpatrick, L., Kim, J.W., Kim, C., Kim, Y., Kirchner, J.H., Kiros, J., Knight, M., Kokotajlo, D., Kondraciuk, Kondrich, A., Konstantinidis, A., Kosic, K., Krueger, G., Kuo, V., Lampe, M., Lan, I., Lee, T., Leike, J., Leung, J., Levy, D., Li, C.M., Lim, R., Lin, M., Lin, S., Litwin, M., Lopez, T., Lowe, R., Lue, P., Makanju, A., Malfacini, K., Manning, S., Markov, T., Markovski, Y., Martin, B., Mayer, K., Mayne, A., McGrew, B., McKinney, S.M., McLeavey, C., McMillan, P., McNeil, J., Medina, D., Mehta, A., Menick, J., Metz, L., Mishchenko, A., Mishkin, P., Monaco, V., Morikawa, E., Mossing, D., Mu, T., Murati, M., Murk, O., Mély, D., Nair, A., Nakano, R., Nayak, R., Neelakantan, A., Ngo, R., Noh, H., Ouyang, L., O'Keefe, C., Pachocki, J., Paino, A., Palermo, J., Pantuliano, A., Parascandolo, G., Parish, J., Parparita, E., Passos, A., Pavlov, M., Peng, A., Perelman, A., Avila Belbute Peres, F., Petrov, M., Oliveira Pinto, H.P., Michael, Pokorny, Pokrass, M., Pong, V.H., Powell, T., Power, A., Power, B., Proehl, E., Puri, R., Radford, A., Rae, J., Ramesh, A., Raymond, C., Real, F., Rimbach, K., Ross, C., Rotsted, B., Roussez, H., Ryder, N., Saltarelli, M., Sanders, T., Santurkar, S., Sastry, G., Schmidt, H., Schnurr, D., Schulman, J., Selsam, D., Sheppard, K., Sherbakov, T., Shieh, J., Shoker, S., Shyam, P., Sidor, S., Sigler, E., Simens, M., Sitkin, J., Slama, K., Sohl, I., Sokolowsky, B., Song, Y., Staudacher, N., Such, F.P., Summers, N., Sutskever, I., Tang, J., Tezak, N., Thompson, M.B., Tillet, P., Tootoonchian, A., Tseng, E., Tuggle, P., Turley, N., Tworek, J., Uribe, J.F.C., Vallone, A., Vijayvergiya, A., Voss, C., Wainwright, C., Wang, J.J., Wang, A., Wang, B., Ward, J., Wei, J., Weinmann, C., Welihinda, A., Welinder, P., Weng, J., Weng, L., Wiethoff, M., Willner, D., Winter, C., Wolrich, S., Wong, H., Workman, L., Wu, S., Wu, J., Wu, M., Xiao, K., Xu, T., Yoo, S., Yu, K., Yuan, Q., Zaremba, W., Zellers, R., Zhang, C., Zhang, M., Zhao, S., Zheng, T., Zhuang, J., Zhuk, W., Zoph, B.: GPT-4 Technical Report (2024). https://doi.org/10.48550/arXiv.2303.08774

[6] Anthropic: System Card: Claude Opus 4 & Claude Sonnet 4. https://www-cdn.anthropic.com/4263b940cabb546aa0e3283f35b686f4f3b2ff47/claude-opus-4-and-claude-sonnet-4-system-card.pdf (2025)

[7] openAI: Codex (2025). https://chatgpt.com/codex/

[8] Li, R., Allal, L.B., Zi, Y., Muennighoff, N., Kocetkov, D., Mou, C., Marone, M., Akiki, C., Li, J., Chim, J., Liu, Q., Zheltonozhskii, E., Zhuo, T.Y., Wang, T.,

Dehaene, O., Davaadorj, M., Lamy-Poirier, J., Monteiro, J., Shliazhko, O., Gontier, N., Meade, N., Zebaze, A., Yee, M.-H., Umapathi, L.K., Zhu, J., Lipkin, B., Oblokulov, M., Wang, Z., Murthy, R., Stillerman, J., Patel, S.S., Abulkhanov, D., Zocca, M., Dey, M., Zhang, Z., Fahmy, N., Bhattacharyya, U., Yu, W., Singh, S., Luccioni, S., Villegas, P., Kunakov, M., Zhdanov, F., Romero, M., Lee, T., Timor, N., Ding, J., Schlesinger, C., Schoelkopf, H., Ebert, J., Dao, T., Mishra, M., Gu, A., Robinson, J., Anderson, C.J., Dolan-Gavitt, B., Contractor, D., Reddy, S., Fried, D., Bahdanau, D., Jernite, Y., Ferrandis, C.M., Hughes, S., Wolf, T., Guha, A., Werra, L., Vries, H.: StarCoder: may the source be with you! (2023). https://doi.org/10.48550/arXiv.2305.06161

[9] Moura, L., Ullrich, S.: The Lean 4 theorem prover and programming language. In: Automated Deduction – CADE 28: 28th International Conference on Automated Deduction, Virtual Event, July 12–15, 2021, Proceedings, pp. 625–635. Springer, Berlin, Heidelberg (2021). https://doi.org/10.1007/978-3-030-79876-5_37

[10] Paulson, L.C.: The Foundation of a Generic Theorem Prover (2000). https://doi.org/10.48550/arXiv.cs/9301105

[11] Development Team, T.: The Coq Proof Assistant (8.19). Zenodo (2024). https://doi.org/10.5281/zenodo.11551307

[12] Megill, N.D.: Metamath: A Computer Language for Mathematical Proofs. Lulu Press, Morrisville, North Carolina (2019). http://us.metamath.org/downloads/metamath.pdf

[13] Owre, S., Rushby, J.M., Shankar, N.: PVS: A prototype verification system. In: International Conference on Automated Deduction, pp. 748–752 (1992). https://doi.org/10.1007/3-540-55602-8_217 . Springer

[14] Page, M.J., McKenzie, J.E., Bossuyt, P.M., Boutron, I., Hoffmann, T.C., Mulrow, C.D., Shamseer, L., Tetzlaff, J.M., Akl, E.A., Brennan, S.E., Chou, R., Glanville, J., Grimshaw, J.M., Hróbjartsson, A., Lalu, M.M., Li, T., Loder, E.W., Mayo-Wilson, E., McDonald, S., McGuinness, L.A., Stewart, L.A., Thomas, J., Tricco, A.C., Welch, V.A., Whiting, P., Moher, D.: The PRISMA 2020 statement: an updated guideline for reporting systematic reviews. BMJ **372**, 71 (2021) https://doi.org/10.1136/bmj.n71

[15] Raayoni, G., Gottlieb, S., Manor, Y., Pisha, G., Harris, Y., Mendlovic, U., Haviv, D., Hadad, Y., Kaminer, I.: Generating conjectures on fundamental constants with the Ramanujan Machine. Nature **590**(7844), 67–73 (2021) https://doi.org/10.1038/s41586-021-03229-4

[16] Davies, A., Veličković, P., Buesing, L., Blackwell, S., Zheng, D., Tomašev, N., Tanburn, R., Battaglia, P., Blundell, C., Juhász, A., Lackenby, M., Williamson, G., Hassabis, D., Kohli, P.: Advancing mathematics by guiding human intuition with AI. Nature **600**(7887), 70–74 (2021) https://doi.org/10.

1038/s41586-021-04086-x

[17] Novikov, A., Vũ, N., Eisenberger, M., Dupont, E., Huang, P.-S., Wagner, A.Z., Shirobokov, S., Kozlovskii, B., Ruiz, F.J.R., Mehrabian, A., Kumar, M.P., See, A., Chaudhuri, S., Holland, G., Davies, A., Nowozin, S., Kohli, P., Balog, M.: AlphaEvolve: A coding agent for scientific and algorithmic discovery. arXiv preprint arXiv:2506.13131 (2025) https://doi.org/10.48550/arXiv.2506.13131

[18] Georgiev, B., Gómez-Serrano, J., Tao, T., Wagner, A.Z.: Mathematical exploration and discovery at scale. arXiv preprint arXiv:2511.02864 (2025) https://doi.org/10.48550/arXiv.2511.02864

[19] Lange, R., Imajuku, Y., Cetin, E.: ShinkaEvolve: Towards open-ended and sample-efficient program evolution. arXiv preprint arXiv:2509.19349 (2025) https://doi.org/10.48550/arXiv.2509.19349

[20] Guo, P., Zhang, Q., Lin, X.: CoEvo: Continual evolution of symbolic solutions using large language models. arXiv preprint arXiv:2412.18890 (2024) https://doi.org/10.48550/arXiv.2412.18890

[21] Anil, R., Dai, A.M., Firat, O., Johnson, M., Lepikhin, D., Passos, A., Shakeri, S., Taropa, E., Bailey, P., Chen, Z., *et al.*: Palm 2 technical report. arXiv preprint arXiv:2305.10403 (2023) https://doi.org/10.48550/arXiv.2305.10403

[22] Hubert, T., Mehta, R., Sartran, L., Horváth, M.Z., Žužić, G., Wieser, E., Huang, A., Schrittwieser, J., Schroecker, Y., Masoom, H., et al.: Olympiad-level formal mathematical reasoning with reinforcement learning. Nature, 1–3 (2025) https://doi.org/10.1038/s41586-025-09833-y

[23] Wagner, A.Z.: Constructions in combinatorics via neural networks. arXiv preprint arXiv:2104.14516 (2021) https://doi.org/10.48550/arXiv.2104.14516

[24] Zhang, C., Song, J., Li, S., Liang, Y., Ma, Y., Wang, W., Zhu, Y., Zhu, S.-C.: Proposing and solving olympiad geometry with guided tree search. Nature Machine Intelligence, 1–12 (2026) https://doi.org/10.1038/s42256-025-01164-x

[25] Dong, K., Ma, T.: STP: Self-play LLM theorem provers with iterative conjecturing and proving. arXiv preprint arXiv:2502.00212 (2025) https://doi.org/10.48550/arXiv.2502.00212

[26] Wang, H., Xin, H., Zheng, C., Li, L., Liu, Z., Cao, Q., Huang, Y., Xiong, J., Shi, H., Xie, E., Yin, J., Li, Z., Liao, H., Liang, X.: LEGO-Prover: Neural theorem proving with growing libraries. arXiv preprint arXiv:2310.00656 (2023) https://doi.org/10.48550/arXiv.2310.00656

[27] Ospanov, A., Farnia, F., Mohit, R.: minif2f-lean revisited: Reviewing limitations and charting a path forward. Advances in Neural Information Processing Systems

38, 79964–80002 (2026). https://proceedings.neurips.cc/paper_files/paper/2025/file/72dad95a24fae750f8ab1cb3dab5e58d-Paper-Conference.pdf

[28] Ying, H., Wu, Z., Geng, Y., Wang, J., Lin, D., Chen, K.: Lean workbook: A large-scale lean problem set formalized from natural language math problems. Advances in Neural Information Processing Systems **37**, 105848–105863 (2024) https://doi.org/10.52202/079017-3357

[29] Johansson, M., Smallbone, N.: Exploring mathematical conjecturing with large language models. In: Proceedings of the 17th International Workshop on Neural-Symbolic Learning and Reasoning (NeSy). CEUR Workshop Proceedings, vol. 3432. CEUR-WS.org, Bonn (2023). https://ceur-ws.org/Vol-3432/paper5.pdf

[30] Alhessi, Y., Einarsdóttir, S.H., Granberry, G., First, E., Johansson, M., Lerner, S., Smallbone, N.: Lemmanaid: Neuro-symbolic lemma conjecturing. In: 2nd AI for Math Workshop @ ICML 2025 (2025). https://openreview.net/forum?id=QS8X04Q0Ov

[31] Lochbihler, A.: Fast machine words in isabelle/hol. In: International Conference on Interactive Theorem Proving, pp. 388–410 (2018). https://doi.org/10.1007/978-3-319-94821-8_23 . Springer

[32] Eberl, M., Klein, G., Lammich, P., Lochbihler, A., Nipkow, T., Paulson, L., Thiemann, R., Traytel, D.: Archive of formal proofs. https://www.isa-afp.org (2024)

[33] Kasaura, K., Onda, N., Oriike, Y., Taniguchi, M., Sannai, A., Sonoda, S.: Discovering new theorems via LLMs with in-context proof learning in Lean. arXiv preprint arXiv:2509.14274 (2025) https://doi.org/10.48550/arXiv.2509.14274

[34] Onda, N., Kasaura, K., Oriike, Y., Taniguchi, M., Sannai, A., Sonoda, S.: LeanConjecturer: Automatic generation of mathematical conjectures for theorem proving. arXiv preprint arXiv:2506.22005 (2025) https://doi.org/10.48550/arXiv.2506.22005

[35] Rabe, M.N., Lee, D., Bansal, K., Szegedy, C.: Mathematical reasoning via self-supervised skip-tree training. In: International Conference on Learning Representations (ICLR) (2020)

[36] Shojaee, P., Nguyen, N.-H., Meidani, K., Farimani, A., Doan, K.D., Reddy, C.K.: LLM-SRBench: A new benchmark for scientific equation discovery with large language models. arXiv preprint arXiv:2504.10415 (2025) https://doi.org/10.48550/arXiv.2504.10415

[37] The mathlib Community: The Lean Mathematical Library. In: Proceedings of the 9th ACM SIGPLAN International Conference on Certified Programs and Proofs. CPP 2020. ACM, New Orleans, LA, USA (2020). https://doi.org/10.

1145/3372885.3373824

[38] Xiong, B., Lv, H., Shan, H., Wang, J., Yang, Z., Zhi, L.: A combinatorial identities benchmark for theorem proving via automated theorem generation. arXiv preprint arXiv:2502.17840 (2025) https://doi.org/10.48550/arXiv.2502.17840

[39] Yin, D., Gao, J.: Generating millions of Lean theorems with proofs by exploring state transition graphs. arXiv preprint arXiv:2503.04772 (2025) https://doi.org/10.48550/arXiv.2503.04772

[40] Sivakumar, J., Borchert, P., Cardenas, R., Lampouras, G.: Conjecturing: An overlooked step in formal mathematical reasoning. arXiv preprint arXiv:2510.11986 (2025) https://doi.org/10.48550/arXiv.2510.11986

[41] Lin, X., Cao, Q., Huang, Y., Yang, Z., Liu, Z., Li, Z., Liang, X.: ATG: Benchmarking automated theorem generation for generative language models. arXiv preprint arXiv:2405.06677 (2024) https://doi.org/10.48550/arXiv.2405.06677

[42] Rosin, C.D.: Using reasoning models to generate search heuristics that solve open instances of combinatorial design problems. arXiv preprint arXiv:2505.23881 (2025) https://doi.org/10.48550/arXiv.2505.23881

[43] Carbone, L.: Advancing mathematics research with generative AI. arXiv preprint arXiv:2511.07420 (2025) https://doi.org/10.48550/arXiv.2511.07420

[44] Cheng, J.: Automated generation of massive reasonable empirical theorems by forward reasoning based on strong relevant logics. arXiv preprint arXiv:2412.12408 (2024) https://doi.org/10.48550/arXiv.2412.12408

[45] Ellenberg, J.S., Fraser-Taliente, C.S., Harvey, T.R., Srivastava, K., Sutherland, A.V.: Generative modeling for mathematical discovery. arXiv preprint arXiv:2503.11061 (2025) https://doi.org/10.48550/arXiv.2503.11061

[46] Rabe, M.N., Lee, D., Bansal, K., Szegedy, C.: Language modeling for formal mathematics (2020) arXiv:2006.04757 [cs.LG]

[47] Du, M., Chen, Y., Wang, Z., Nie, L., Zhang, D.: Large language models for automatic equation discovery of nonlinear dynamics. Physics of Fluids **36**(9) (2024) https://doi.org/10.1063/5.0224297

[48] Saidi, H., Jha, S., Sahai, T.: math-PVS: A large language model framework to map scientific publications to PVS theories. arXiv preprint arXiv:2310.17064 (2023) https://doi.org/10.48550/arXiv.2310.17064

[49] Chuharski, J., Collins, E.R., Meringolo, M.: Mining math conjectures from LLMs: A pruning approach. arXiv preprint arXiv:2412.16177 (2024) https://doi.org/10.48550/arXiv.2412.16177 . NeurIPS 2024 MATH-AI Workshop

[50] Nagda, A., Raghavan, P., Thakurta, A.: Reinforced generation of combinatorial structures: Hardness of approximation. arXiv preprint arXiv:2509.18057 (2025) https://doi.org/10.48550/arXiv.2509.18057

[51] Abdali, S., Goksen, C., Amizadeh, S., Koishida, K.: Self-reflecting large language models: A Hegelian dialectical approach. arXiv preprint arXiv:2501.14917 (2025) https://doi.org/10.48550/arXiv.2501.14917

[52] Zhao, A., Wu, Y., Yue, Y., Wu, T., Xu, Q., Yue, Y., Lin, M., Wang, S., Wu, Q., Zheng, Z., Huang, G.: Absolute Zero: Reinforced self-play reasoning with zero data. arXiv preprint arXiv:2005.03335 (2025) https://doi.org/10.48550/arXiv.2505.03335

[53] Trinh, T., Wu, Y., He He, Q., Luong, T.: Solving olympiad geometry without human demonstrations. Nature (625), 476–482 (2024) https://doi.org/10.1038/s41586-023-06747-5

[54] Liang, Z., Song, L., Li, Y., Yang, T., Zhang, F., Mi, H., Yu, D.: Towards solving more challenging imo problems via decoupled reasoning and proving. arXiv preprint arXiv:2507.06804 (2025) https://doi.org/10.48550/arXiv.2507.06804

[55] Luu, R., Buehler, M.: Bioinspiredllm: Conversational large language model for the mechanics of biological and bio-inspired materials. arXiv preprint arXiv:2309.08788 (2023) https://doi.org/10.48550/arXiv.2309.08788

[56] Ghafarollahi, A., Buehler, M.: Sciagents: Automating scientific discovery through bioinspired multi-agent intelligent graph reasoning. AdvancedMaterials **37**(22) (2025) https://doi.org/10.1002/adma.202413523

[57] Sprueill, H., Edwards, C., Olarte, M., Sanyal, U., Ji, H., Choudhury, S.: Monte carlo thought search: Large language model querying for complex scientific reasoning in catalyst design. arXiv preprint arXiv:2310.14420 (2023) https://doi.org/10.48550/arXiv.2310.14420

[58] Tang, J., Xia, L., Li, Z., Huang, C.: AI-researcher: Autonomous scientific innovation. arXiv preprint arXiv:2505.18705 (2025) https://doi.org/10.48550/arXiv.2505.18705

[59] Ban, T., Chen, L., Lyu, D., Wang, X., Chen, H.: Causal structure learning supervised by large language model. arXiv preprint arXiv:2311.11689 (2025) https://doi.org/10.48550/arXiv.2311.11689

[60] Luo, L., Ju, J., Xiong, B., Li, Y.-F., Haffari, G., Pan, S.: Chatrule: Mining logical rules with large language models for knowledge graph reasoning. In: Advances in Knowledge Discovery and Data Mining: 29th Pacific-Asia Conference on Knowledge Discovery and Data Mining, PAKDD 2025, Sydney, NSW, Australia, June 10-13, 2025, Proceedings, Part II, pp. 314–325. Springer, Berlin, Heidelberg

(2025). https://doi.org/10.1007/978-981-96-8173-0_25

[61] Hao, H., Zhang, K., Xiong, M.: Dynamic models of neural population dynamics. bioRxiv (2024) https://doi.org/10.1101/2024.10.04.616750

[62] Kastrin, A., Cestnik, B., Lavrač, N.: Recent advances and future directions in literature-based discovery. arXiv preprint arXiv:2506.12385 (2025) https://doi.org/10.48550/arXiv.2506.12385

[63] Li, R., Zhang, C., Mao, S., Zhou, X., Yin, F., Theodoridis, S., Zhang, Z.: LLM4Laser: Large language models automate the design of lasers. arXiv preprint arxiv:2104.12145 (2025) https://doi.org/10.48550/arXiv.2104.12145

[64] Bolibar, J., Sapienza, F., Maussion, F., Lguensat, R., Wouters, B., Pérez, F.: Universal differential equations for glacier ice flow modelling. Geoscientific Model Development **16**(22), 6671–6687 (2023) https://doi.org/10.5194/gmd-16-6671-2023

[65] Chen, Q., Kawashima, H.: Sentiment-aware stock price prediction with transformer and LLM-generated formulaic alpha. arXiv preprint arXiv:2508.04975 (2026) https://doi.org/10.48550/arXiv.2508.04975

[66] Narin, A.E.: Math problem solving: Enhancing large language models with semantically rich symbolic variables. In: Valentino, M., Ferreira, D., Thayaparan, M., Freitas, A. (eds.) Proceedings of the 2nd Workshop on Mathematical Natural Language Processing @ LREC-COLING 2024, pp. 19–24. ELRA and ICCL, Torino, Italia (2024). https://aclanthology.org/2024.mathnlp-1.3/

[67] Romera-Paredes, B., Schmid, C., Veličković, P., Tuyls, K., Tacchetti, A., Vinyals, O., Hassabis, D.: Automated discovery of new cap-set constructions with funsearch. Nature **627**, 125–132 (2024) https://doi.org/10.1038/s41586-024-XXXXX

[68] Xia, P., Zeng, K., Liu, J., Qin, C., Wu, F., Zhou, Y., Xiong, C., Yao, H.: Agent0: Unleashing self-evolving agents from zero data via tool-integrated reasoning. arXiv preprint arxiv:2511.16043 (2025) https://doi.org/10.48550/arXiv.2511.16043

[69] Wang, R., Zelikman, E., Poesia, G., Pu, Y., Haber, N., Goodman, N.D.: Hypothesis search: Inductive reasoning with language models. arXiv preprint arxiv:2309.05660 (2024) https://doi.org/10.48550/arXiv.2309.05660