The Symbiosis of Formal Methods and Artificial Intelligence

Jonathan P. Bowen *

Posted Date: 4 March 2026

doi: 10.20944/preprints202603.0373.v1

Keywords: artificial intellience; formal methods; generative AI; research aids; software engineering

 doi:10.20944/preprints202603.0373.v1

Article

The Symbiosis of Formal Methods and Artificial Intelligence

Jonathan P. Bowen [1,2,*]

[1] School of Computer Science and Digital Technologies, London South Bank University, UK; jonathan.bowen@lsbu.ac.uk
[2] RISE, Southwest University, Chongqing, China

Highlights

The paper considers the effect of developments in Artificial Intelligence (AI), especially Generative AI (GenAI), on the use of formal methods in software engineering.

What are the main findings?

- Formal methods are precise but difficult to use.
- Generative AI is imprecise but relatively easy to use.
- A combination of these approaches promises benefits for software engineering.

What are the implications of the main findings?

- Formal methods may become easy to use with AI.
- AI may become more exact with the use of formal methods.
- Rigourous sotware engineering could become easier in practice.

Abstract

Formal methods are software engineering approaches with a rigorous mathematical basis that can be used in helping to ensure the correctness of software systems, especially where safety or security is critical. Artificial Intelligence (AI) has developed very rapidly in the area of Generative AI (GenAI), where questions can be answered with increasingly impressive but with potentially unreliable and variable results. This paper surveys research in integrating the two approaches in a synergistic manner. Traditionally, such explorations have required significant manual efforts in searching for and evaluating existing research. However, most relevant publications are now accessible online, and AI tools are increasingly good at answering research questions with more and more reliability. This paper takes the approach of using GenAI to evaluate research questions on combining formal methods and AI-related techniques. The paper assesses the usefulness and validity of these results. With recent improvements in AI search tools, the approach is now a useful aid to researchers, significantly reducing the time needed to survey existing research, while always needing human checking by an expert. In addition, the combination of formal methods and AI approaches looks to be an interesting and beneficial research area with potential industrial-scale applications in the future.

Keywords: artificial intellience; formal methods; Generative AI; research aids; software engineering

1. Introduction

Alan Turing arguably wrote the first paper on formal methods [64], using mathematically based approaches to software engineering, at a 1949 conference in Cambridge [53,61]. A year later in 1950, he wrote a seminal article on machine intelligence [62], predicting the advent of what became known as Artificial Intelligence (AI). Since then, there have been claims and counterclaims about formal methods. Tool development has been slow, but aided by Moore's Law with the increasing power of computers. Although formal methods are not widespread in practical usage at a heavyweight level, their influence has crept into software engineering practice to the extent that they are no longer necessarily called

 doi:10.20944/preprints202603.0373.v1

formal methods in their use. In addition, in areas where safety and security are important, with the increasing use of computers in such applications, formal methods are a viable way to improve the reliability of such software-based systems. Their use in hardware, where a mistake can be very costly, is also important.

This paper presents the use of formal methods in the light of developing directions regarding their use with the aid of Generative AI (GenAI). It uses an interactive GenAI approach to survey the state of the art in combining formal methods and AI approaches. The former has the advantage of mathematical exactitude, whereas the latter, while inexact, is proving increasingly good at apparently serendipitous discoveries.

1.1. Formal Methods Background

While the logical foundations of formal methods can be traced back to Aristotle's logic and the desire to formalize reasoning, the application to computing has distinct 20th-century roots. In mathematics, it is typical to have simply stated theorems, by with potentially very deep and complex proofs. Consider Fermat's Last Theorem (c. 1637) – $a^n + b^n \neq c^n$ for integer $n > 2$ – postulated by the French mathematician Pierre de Fermat (1607–1665). This was finally proved by Andrew Wiles around 358 years later, in 1994–5, not a timescale acceptable in software development! Instead, in software verification, there are complicated specifications, potentially thousands of pages, and long programs with millions of lines of code, but with relatively shallow proofs compared to those developed by professional mathematicians.

Arguably, the first paper on formal methods was written by Alan Turing in 1949, titled "Checking a Large Routine" [61]. In this seminal work, presented at the EDSAC Inaugural Conference, Turing introduced the concept of making assertions about the various states that a machine can reach. He posited that the checker of a routine has to verify the initial and final conditions, assertions within the program flow, and that the process terminates.

Turing utilized a flow diagram with properties attached to edges, anticipating the later work of Floyd and Hoare by decades. As noted by Morris and Jones in their analysis of Turing's work, had Turing lived longer, the field of program proving might have developed much more rapidly, rather than requiring the ideas to be rediscovered in the late 1960s.

Reading Turing's paper now, it uses many of the terms that are common in formal methods today, such as "verification", "assertions", and even the "dashed" after states used in the formal Z notation [34].

But what of Turing's influence on program proving? Adriaan "Aad" van Wijngaarden (1916–1987), a Dutch mathematician and computer scientist, and an early advocate of a mathematical approach to computing, was at the Cambridge meeting, where Turing presented his ideas, but there is no obvious influence on van Wijngaarden.

Robert Floyd rediscovered ideas similar to those of Turing, published in 1967 [41]. Tony Hoare developed these further, published in his classic 1969 paper presenting what has become known as "Hoare Logic" [49]. Had Turing lived longer, perhaps formal methods (in particular, program proving) would have developed more rapidly and with greater continuity, rather than being rediscovered.

One of the earliest books with "formal methods" in its title is a 1962 volume by the Dutch philosopher and logician, Evert Willem Beth (1908–1964) [32]. The full title is *Formal Methods: An Introduction to Symbolic Logic and to the Study of Effective Operations in Arithmetic and Logic*. Although this book is not about software, it does use the term "formal methods" with a meaning that is recognizable by the formal methods community of today. The term "formal methods" was established in its computer science context by the late 1970s, as the next more rigorous stage from structured design, establishing a mathematical basis. Despite advances, a gap remains between theory and practice. In the 1970s, Christopher Strachey, a pioneer in denotational semantics, a particular approach to formalizing software, observed that practical work is often "unsound and clumsy" due to a lack of understanding of fundamental design principles, while theoretical work is often "sterile" because it lacks contact with real computing [59].

 doi:10.20944/preprints202603.0373.v1

By the 1990s, formal methods had established a presence but were surrounded by misconceptions regarding their difficulty and utility. In 1990, Anthony Hall proposed seven "myths" of formal methods, commonly held beliefs about the approach that are inaccurate in practice [46]. Further myths were proposed in 1995 [36], including the ideas that formal methods delay the development process, lack tool support, replace traditional engineering design, only apply to software, or are simply not required. A persistent fallacy was the "all or nothing" approach – the belief that one must be dogmatic. In reality, a related paper on the "Ten Commandments of Formal Methods" [35] suggested a pragmatic approach. For example, "Thou shalt formalize but not over-formalize" and "Thou shalt guestimate costs."

In the 1995 book *Applications of Formal Methods*, a number of case studies using formal methods were presented using different formal methods [47]. A follow-on book in 1999 [48] provided further examples. A 2001 book [42] presented the process of developing a formal specification for a unified case study by explicitly showing the questions raised when developing such a specification using a wide selection of different formal methods. This contrasts with many presentations where only the final specification of a system is provided, with little information on how this was derived. Further formal methods case study books have been published [33,44].

Formal methods involve formal specification and (optionally) proof, covering:

Validation: helping to establish a correct specification; and

Verification: establishing a correct implementation with respect to a formal specification.

It should be noted that the use of proof in a software engineering context is an outlier compared to most engineering disciplines. Engineers typically "calculate", for example, using mathematical formulae developed and "proved" previously (e.g., using Maxwell's equations).

There are a variety of ways to apply formal methods to computer-based systems, including:

Abstract Interpretation: approximating program behaviour to prove correctness or detect errors.

Model-Based Testing: generating test cases from a formal model.

Model Checking: exhaustively verifying system behaviour against a formal specification.

Proof Assistants: tools for interactively constructing and verifying mathematical proofs.

Refinement: systematically refining a high-level specification into a correct implementation.

Static Analysis: analyzing program code meaning to detect errors or enforce constraints.

Verification: proving the correctness of a program using logical inference rules.

There are also different levels at which formal methods can be used:

1. **Formal Specification:** This is applicable to the system requirements. Formal analysis and proofs are optional. The specification can be used to aid testing. It is the most cost-effective approach using formal methods.
2. **Formal Verification:** The implemented program is produced in a more mathematically formal way. Typically, the approach uses proof or refinement based on a formal specification. This is significantly more costly than just using formal specification.
3. **Theorem Proving:** This involves the use of a theorem prover tool. The approach allows formal machine-checked proofs. A proof of the entire system is possible, but scaling is difficult. The approach can be very expensive and hard.

Formal methods have found their specific niche in the development of systems where safety or security is critical. This has sometimes been driven by the recommendation or even mandating of the use of formal methods in systems at the most critical levels in standards [37], often where human lives are at risk in the case of failure.

 doi:10.20944/preprints202603.0373.v1

1.2. Formal Methods and AI

With the recent developments in Generative AI (GenAI), there has been an increasing interest in the possibility of combining the precision of formal methods with the, at least partially, automated possibility of using GenAI to aid in the generation of proofs during verification using a formal approach. This is a rapidly developing area. This paper aims to survey the current state of the art in this area, using a novel approach of utilizing GenAI itself to aid in exploring the current research. In Section 2, we introduce the methodological approach and in Section 3 we present the results of posing three research questions, with comments on each of the publications returned by Google Scholar Labs. We discuss these results and also general developments in combining formal methods and AI in Section 4, followed by a brief conclusion in Section 5.

2. Methodology

Traditionally, a survey of academic papers in a particular area has required significant manual effort. The systematic literature review process has typically required multiple authors to be comprehensive. This has been the case in the software engineering domain, for example [38]. In the context of this traditional approach, the best search strategies have been explored [40]. Guidelines for conducting systematic mapping studies for software engineering in particular have been developed and later revised [56,57].

However, the rapid development of Generative AI (GenAI) has enabled a more interactive approach to answering research questions with respect to surveying academic papers. For some time, since its launch in 2004, Google Scholar (https://scholar.google.com) has provided good access and search facilities for academic publications, especially for computer science, where most research papers are accessible online. Google has now added Google Scholar Labs to its Google Scholar search facility [45], launched by Google on November 18, 2025, as an experimental AI-powered additional feature for the Google Scholar platform. This allows natural language questions to be posed, returning what the AI-driven facility judges to be the most relevant publications, together with a summary description of their relevance. In this paper, as an experiment, we pose three research questions on formal methods and Artificial Intelligence using the Google Scholar Labs facilities. We present the results of these questions, together with an expert judgement of their usefulness in practice in Section 3.

3. Results

In this section, we consider three research questions posed to Google Scholar Labs [45] and the first ten entries of relevant publications that are returned. The AI-generated summary information provided by Google is included, together with a critique of its usefulness, especially in the context of the research questions.

Each publication that is listed starts with basic information adapted from the results provided by Google Scholar Labs, including a citation for the publication included in References 1-35. The Google Scholar Labs summary format uses a standard style, with an overview sentence, followed by two bullet points, each with a brief description of aspects covered in the publication.

Question 1: Is Artificial Intelligence useful for formal methods?

This general research question was first posed to Google Scholar Labs, with the following results for the top ten publications returned.

Publication 1.1

[BOOK] *Formal Methods in Artificial Intelligence* [1].

A Ramsay – 1988 – books.google.com

Explains that Artificial Intelligence turns to formal logic for powerful and well-behaved knowledge representation languages.

- Enriching AI with Logic: Aims to show how Artificial Intelligence can be enriched by

 doi:10.20944/preprints202603.0373.v1

incorporating recent advances in formal logic, including extensions of classical logic for reasoning about time, knowledge, or uncertainty.

• Borrowing Formal Techniques: Details that techniques from formal logic have been borrowed by AI researchers, often for applications outside their original context, resulting in useful and constructive work.

Cited by 118

This is an extensive and well-cited book, but rather dated for current relevance, having been published in 1988 [1]. The book's title fits very well with the research question, but its age means that it is now largely of historical interest.

Publication 1.2

[PDF] researchgate.net

Are formal methods applicable to machine learning and artificial intelligence? [2].

M Krichen, A Mihoub, MY Alzahrani, et al. – Proceedings, 2022 – ieeexplore.ieee.org

Examines state-of-the-art formal methods for the verification and validation of machine learning (a subset of AI) systems.

• Verifying ML Models: Reviews research works dedicated to the verification of specific machine learning models, such as support vector machines and decision tree ensembles.

• Verifying ML Phases: Summarizes formal verification efforts for the data preparation, training phases, and the resulting trained machine learning models, including neural networks.

Cited by 128

This conference paper considers the use of formal methods in machine learning and AI to aid in verifying their correctness. Its publication in 2022 means that it was written before ChatGPT (which was launched on November 30, 2022) became generally available, so it predates the recent GenAI revolution. That said, it is well-cited and poses a prescient question.

Publication 1.3

[PDF] zulipchat.com

A promising path towards autoformalization and general artificial intelligence [3].

C Szegedy – Proceedings, 2020 – Springer

Argues that autoformalization, an AI system that converts natural language into machine-verifiable formalization, is a promising direction for creating automated mathematical reasoning systems.

• Automating Formal Proofs: Discusses the potential of autoformalization to automate the creation of computer verifiable proofs, which traditionally require significant human effort.

• AI Techniques: Explains that advances in machine learning techniques, such as neural networks and reinforcement learning, are paving the way for strong automated reasoning and formalization systems.

Cited by 94

This 2020 conference paper predates even ***Publication 1.2*** and discusses the potential of AI aiding computer-verifiable proofs and automated reasoning, both useful in computer-aided formal methods. However, like ***Publication 1.2***, it looks at future directions, which are now overtaken by more recent developments.

Publication 1.4

Computer theorem proving and artificial intelligence [4].

H Wang – Computation, Logic, Philosophy: A Collection of Essays, 1990 – Springer

 doi:10.20944/preprints202603.0373.v1

Discusses the use of computer programs for theorem proving, which is a key component of formal methods, and its relationship to Artificial Intelligence (AI).

• Early Program Results: Describes early programs that applied formal logic to prove a large number of theorems from Principia Mathematica in the propositional and predicate calculus, demonstrating the utility of AI for formal methods.

• Expert Systems for Proof: Presents a plea for 'expert systems' in the context of theorem proving, suggesting that partial, efficient strategies should be emphasized over general algorithms for proving theorems on a machine.

Cited by 15

This 1990 philosophical chapter, while having some relevance, is now largely of historical interest, looking back to mathematical theorems in Alfred North Whitehead and Bertrand Russell's 1910–1913 three-volume *Principia Mathematica* [63] and traditional expert systems (first introduced around 1965 by Edward Feigenbaum *et al.*), a decision-making form of AI that has been superseded by GenAI.

Publication 1.5

Automatic B-model repair using model checking and machine learning [5].

CH Cai, J Sun, G Dobbie – Automated Software Engineering, 2019 – Springer

Proposes B-repair, an approach for automated repair of faulty models written in the B formal specification language, which uses machine learning techniques to estimate the quality of suggested repairs.

• AI and Model Checking Integration: Combines model checking and machine learning-guided techniques to automatically find and repair faults during the design of software systems.

• AI for Repair Quality Estimation: Uses machine learning techniques to learn the features of state transitions, which helps in ranking suggested repairs and selecting the best one for revising the formal model.

Cited by 22

This 2019 journal paper is interesting because it uses a specific formal methods approach, namely the specification language of the B-Method [30]. However, although it has received some citations, the ideas in this paper are largely subsumed by GenAI developments.

Publication 1.6

[PDF] acm.org

Formal verification of neural network controlled autonomous systems [6].

X Sun, H Khedr, Y Shoukry – Proceedings, 2019 – dl.acm.org

Explores the safety and reliability challenges arising from the increasing use of Artificial Intelligence (AI) agents, such as deep neural network models, in controlling physical and mechanical systems.

• Formal Verification of NNs: Focuses on applying formal verification techniques to neural network models, including those used in autonomous systems, to provide safety guarantees that testing and falsification methods lack.

• NN-Controlled Robot Safety: Considers the formal verification of the safety of an autonomous robot equipped with a Neural Network controller that processes sensor data (LiDAR images) to produce control actions, demonstrating a practical application of formal methods to an AI-controlled system.

Cited by 208

 doi:10.20944/preprints202603.0373.v1

This 2019 conference paper is very well cited and considers the use of formal methods to verify neural networks, rather than using AI to aid formal methods. For this reason, its findings are likely to remain valid; hence, a reason for its high citations, even though it predated recent GenAI developments.

Publication 1.7

[PDF] arxiv.org

The fusion of large language models and formal methods for trustworthy AI agents: A roadmap [7].

Y Zhang, Y Cai, X Zuo, X Luan, K Wang, et al. – *arXiv* preprint, 2024 – arxiv.org

Highlights that Large Language Models (LLMs), a form of Artificial Intelligence, can significantly enhance the usability, efficiency, and scalability of existing formal methods (FM) tools.

• Advancing Formal Methods Tasks: Explores how LLMs can drive advancements in formal methods by automating the formalization of specifications, leveraging LLMs to enhance theorem proving, and developing intelligent model checkers.

• AI as Intelligent Assistants: Details that AI, specifically LLMs, can act as intelligent assistants to formal methods by automating tedious tasks and enhancing the capabilities of existing FM algorithms for verification processes.

Cited by 31

This is the first paper to be returned that dates within the recent period of GenAI development (2024). It is also published on the em arXiv paper archive website run by Cornell University. This was launched in 1991 and is now very well established for computer science-related publications, as well as other scientific and mathematical fields. Papers are not formally peer reviewed, but are vetted for relevance. It thus provides a fast means of publication, which is very attractive in the rapid-paced GenAI area. This is the first paper returned that is very relevant to the current advances in using AI to aid formal methods by providing an "intelligent" assistant for their use. The paper highlights efficiency, scalability, and usability, which are all significant issues in the application of formal methods.

Publication 1.8

[PDF] neurips.cc

MLFMF: data sets for machine learning for mathematical formalization [8].

A Bauer, et al. – Proceedings, 2023 – proceedings.neurips.cc

Introduces MLFMF, a collection of data sets for benchmarking recommendation systems that use machine learning to support the formalization of mathematics with proof assistants.

• Evaluates ML Approaches: Reports baseline results using standard graph and word embeddings, tree ensembles, and instance-based learning algorithms to evaluate machine learning approaches to formalized mathematics.

• Aiding Formalization Efforts: Describes how machine learning methods are used to address premise selection, such as recommending theorems useful for proving a given statement, and how various neural networks and reinforcement learning are applied to automated theorem proving.

Cited by 13

This 2023 conference paper was published during the early days of the recent GenAI developments. It uses formalization but is less relevant to the application of forma methds in a software engineering context. It is not very well cited and could be considered of marginal interest with respect to the posed research question.

Publication 1.9

[PDF] units.it

Machine learning methods in statistical model checking and system design – tutorial [9].

 doi:10.20944/preprints202603.0373.v1

L Bortolussi, D Milios, G Sanguinetti – Proceedings, 2015 – Springer

Reviews methodologies that employ machine learning concepts, specifically Bayesian methods and Gaussian Processes, for model checking and system design/parameter synthesis against stochastic dynamical models.

• Convergence of Ideas: Focuses on the increasingly fertile convergence of ideas from machine learning and formal modelling, highlighting it as a growing research area.

• Bayesian Methods for Efficiency: Details the use of powerful methodologies from Bayesian machine learning and reinforcement learning to obtain efficient algorithms for statistical model checking and parameter synthesis.

Cited by 13

This is a tutorial paper from 2015, and is thus rather out of date to be of current research interest. It considers formal modelling, rather than the use of formal methods in a software engineering context, and is not particularly well cited in any case.

Publication 1.10

Machine learning and formal concept analysis [10].

SO Kuznetsov – Proceedings, 2004 – Springer

Explains that a model of learning from positive and negative examples, a core aspect of Artificial Intelligence/Machine Learning, can be naturally described using the means of Formal Concept Analysis (FCA), a formal method.

• Describing AI Models: Demonstrates the usefulness of Formal Concept Analysis by applying its language to give natural descriptions of standard Machine Learning models, including version spaces and decision trees.

• Comparing Approaches: Suggests that using the language of FCA allows for the comparison of several machine learning approaches and the employment of standard FCA techniques within the domain of machine learning.

Cited by 206

Dating from 2004, this paper, although well cited, is now largely of historical interest. Although Google Scholar Labs considers Formal Concept Analysis to be a "formal method", it is really related to a concept hierarchy forming a formal ontology from a collection of objects together with their properties. This has limited relevance to formal methods for development in a software engineering context.

Summary:

The most relevant publication returned by the first research question is ***Publication 1.7***. Many of the others predate recent GenAI developments. They may be of historical interest, but if researching current trends, they are less useful.

Question 2: Can Generative AI be used to aid formal methods?

Next, this more specific research question was posed to Google Scholar Labs, to gauge the usefulness of using GenAI with formal approaches to software engineering. This improved the relevance of publications that were returned.

Publication 2.1

[PDF] fiu.edu

Dehallucinating large language models using formal methods guided iterative prompting [11].

S Jha, SK Jha, P Lincoln, et al. – Proceedings, 2023 – ieeexplore.ieee.org

 doi:10.20944/preprints202603.0373.v1

Exploits the dialog capability of large language models (LLMs) to iteratively steer them toward responses consistent with a correctness specification using formal methods.

• Self-Monitoring Architecture: Describes a self-monitoring and iterative prompting architecture that uses formal methods to automatically detect errors and inconsistencies in LLM responses, focusing on applications in autonomous systems.

• Formal Methods Integration: Presents a loose coupling between the LLM and a counterexample generator, allowing the use of different formal methods, such as model checking and SMT solving, to achieve a high-assurance solution.

Cited by 106

This 2023 conference paper covers the use of formal methods to aid Large Language Models (LLMs) for AI, rather than the reverse, with the aim of avoid hallucination. Nevertheless, it is highly cited and of potential interest.

Publication 2.2

[PDF] arxiv.org

Generative language modeling for automated theorem proving [12].

S Polu, I Sutskever – *arXiv* preprint, 2020 – arxiv.org

Explores the application of transformer-based language models, a form of Generative AI, to the domain of automated theorem proving, a key aspect of formal methods.

• GPT-f Proof Assistant: Presents an automated prover and proof assistant, GPT-f, which uses generative language modeling for the Metamath formalization language.

• Practical Contribution of AI: Contributed new short proofs that were adopted by the formal mathematics community, demonstrating the practical utility of a deep-learning based system in this field.

Cited by 409

This 2020 preprint paper is early in recent GenAI developments, but is very well cited. It covered using GenAI to aid automated theorem proving, so it is relevant to formal methods-style verification with computer-based proof tools. However, it is more oriented to proofs of general mathematical theorems using the Metamath proof assistant tool, rather than proofs relating to formal methods development approaches for software development. The former tend to be deeper but more concisely stated, whereas proofs for software tend to be shallower, but much more extensive due to the complexity of software in practice.

Publication 2.3

[PDF] arxiv.org

The fusion of large language models and formal methods for trustworthy AI agents: A roadmap [7].

Y Zhang, Y Cai, X Zuo, X Luan, K Wang, et al. – *arXiv* preprint, 2024 – arxiv.org

Highlights how the advanced learning capabilities and adaptability of Large Language Models (LLMs), a form of Generative AI, can significantly enhance the usability, efficiency, and scalability of existing Formal Methods (FM) tools.

• Driving FM Advancements: Examines how LLMs can drive advancements in existing formal methods and tools, focusing on utilizing LLMs for automated formalization of specifications, enhancing theorem proving, and developing intelligent model checkers.

• Automating Verification Tasks: Proposes using LLMs to make formal methods more efficient and effective by serving as intelligent assistants that automate tedious tasks, such as generating structured code and symbolic representations for verification processes.

Cited by 31

 doi:10.20944/preprints202603.0373.v1

This paper was provided as ***Publication 1.7*** for the first research question posed earlier. It is interesting to note the difference in the summary by Google Scholar Labs. The meaning is largely the same, but the wording is different, presumanly due to the random element in the generation of output from GenAI algorithms. As mentioned previously, it is very relevant to the area of interest for both research questions, so it is good to see that it is returned in both cases. With the second more focussed research question, it is returned earlier in the list (third rather than seventh).

Publication 2.4

[PDF] arxiv.org

Goedel-Prover: A frontier model for open-source automated theorem proving [13].

Y Lin, S Tang, B Lyu, J Wu, H Lin, K Yang, et al. – *arXiv* preprint, 2025 – arxiv.org

> Introduces an open-source language model, Goedel-Prover, that achieves state-of-the-art performance in automated formal proof generation for mathematical problems, directly demonstrating the use of generative AI in formal methods.
>
> • Training LLMs for Formalization: Explains the process of training Large Language Models (LLMs) to convert natural language math problems into equivalent formal statements in Lean 4, which is a key step in applying AI to formal methods.
>
> • Developing Formal Proof Datasets: Demonstrates training a series of provers to develop a large dataset of formal proofs, showcasing how generative AI can overcome the scarcity of formalized mathematical data.
>
> Cited by 75

This paper presents Version 1 of the Goedel-Prover automated theorem prover. Although published in 2025, it had already received 75 citations in early 2026. Being published on the *arXiv* archive no doubt contributed to the speed of recognition by authors of other publications. As for ***Publication 2.3***, it is also relevant to the research question, although it concentrates more on proofs for mathematical theorems, like ***Publication 2.2***, rather than software verification proofs. It is the first publication to cover the Lean proof assistant and its associated functional programming language (https://lean-lang.org), originally introduced in 2013. This paper uses Lean 4 language, released in 2021. Recently, Lean has become central in investigations of aiding mathematical proofs using AI. The Goedel-Prover (https://goedel-lm.github.io) covered in this paper is an open-source language model aimed at automated formal proof generation for mathematical problems using the Lean 4 language.

Publication 2.5

[PDF] arxiv.org

Goedel-Prover-V2: Scaling formal theorem proving with scaffolded data synthesis and self-correction [14]

Y Lin, S Tang, B Lyu, Z Yang, et al. – *arXiv* preprint, 2025 – arxiv.org

> Introduces Goedel-Prover-V2, an open-source language model series that achieves a new state-of-the-art in automated theorem proving (a key component of formal methods).
>
> • Generating Formal Proofs: Demonstrates the use of a generative AI model to create machine-verifiable proofs in formal languages like Lean, which is essential for formal methods.
>
> • Compiler-Guided Proof Correction: Presents a method where the model iteratively revises its proofs by utilizing feedback from the Lean compiler (verifier-guided self-correction), integrating verification feedback into the proof generation process.
>
> Cited by 29

This paper relates to that in the previous ***Publication 2.4*** [13], with overlapping authors and also on the *arXiv* archive. It covers Version 2 of the Goedel-Prover automated theorem prover, including use of the Lean 4 language. Even this had already received 29 citations by early 2026. It looks like developments

 doi:10.20944/preprints202603.0373.v1

in the Goedel-Prover will be worth watching with respect to the possibilities of proofs that could be relevant to formal methods approaches.

Publication 2.6

[PDF] arxiv.org

DeepSeek-Prover-V1.5: Harnessing proof assistant feedback for reinforcement learning and Monte-Carlo tree search [15].

H Xin, ZZ Ren, J Song, Z Shao, et al. – *arXiv* preprint, 2024 – arxiv.org

Introduces DeepSeek-Prover-V1.5, an open-source language model specifically designed for theorem proving in Lean 4, demonstrating a direct application of generative AI in formal methods.

• Influences Mathematical Reasoning: Discusses how recent advancements in large language models have significantly influenced mathematical reasoning and theorem proving in artificial intelligence, confirming the relevance of generative AI in this domain.

• Employs Proof Generation Strategies: Presents two common strategies language models use in formal theorem proving: proof-step generation and whole-proof generation, detailing mechanisms for utilizing generative AI in creating formal proofs.

Cited by 42

This 2024 paper covers the DeepSeek-Prover proof assistant, also covered later in a related paper under ***Publication 2.9*** [16]. DeepSeek is a Chinese AI company that was founded in 2023. Its Large Language Models have significantly lower training costs than others like OpenAI's GPT-4 and Meta's Llama 3.1, for example. DeepSeek-Prover is an open-source AI model for mathematical theorem proving and formal reasoning. It is already available as Version 2 (https://prover-v2.com) in this fast-moving field. As for ***Publication 2.5***, it uses the Lean 4 language and is aimed at mathematical theorems in general. Nevertheless, this proof assistant could be one to watch for developments in the future with respect to use for software verification.

Publication 2.7

[PDF] arxiv.org

Verification of image-based neural network controllers using generative models [17].

SM Katz, AL Corso, et al. – Journal, 2022 – arc.aiaa.org

Proposes a method to address the scaling challenges of formal methods for image-based neural network controllers by training a Generative Adversarial Network (GAN) to map states to plausible input images.

• Reducing Input Dimension: Shows that concatenating the generative network with the control network results in a network with a low-dimensional input space, enabling the use of existing closed-loop verification tools to obtain formal guarantees.

• Applying to Safety Guarantees: Applies the approach to provide safety guarantees for an image-based neural network controller for an autonomous aircraft taxi problem.

Cited by 85

This 2025 journal paper considers scaling issues in using formal methods for neural network controllers for image-based sensors. This is a relatively specialist area, but solutions for formal methods scaling challenges are always of interest.

Publication 2.8

[PDF] arxiv.org

Kimina-Prover Preview: Towards large formal reasoning models with reinforcement learning [18].

H Wang, M Unsal, X Lin, M Baksys, et al. – *arXiv* preprint, 2025 – arxiv.org

 doi:10.20944/preprints202603.0373.v1

Introduces Kimina-Prover Preview, a large language model designed for formal theorem proving, demonstrating its application in Lean 4 proof generation.

• Emulates Human Reasoning: Employs a reasoning-driven exploration paradigm and a structured formal reasoning pattern to emulate human problem-solving strategies in the Lean 4 proof assistant, iteratively generating and refining proof steps.

• State-of-the-Art Performance: Achieves state-of-the-art results on the miniF2F benchmark for automated theorem proving, confirming the effectiveness of generative AI models in formal verification.

Cited by 74

As with ***Publication 2.4***, ***Publication 2.5***, and ***Publication 2.6***, this 2025 paper makes use of the Lean 4 language. It introduced the Kimina-Prover Preview large language model for theorem proving. This relatively new proof tool is already well cited. As such, it looks like it will be worth monitoring for its developments and possible use of formal methods in the future.

Publication 2.9

[PDF] arxiv.org

DeepSeek-Prover: Advancing theorem proving in LLMs through large-scale synthetic data [16].

H Xin, D Guo, Z Shao, Z Ren, Q Zhu, et al. – *arXiv* preprint, 2024 – arxiv.org

Introduces an approach to generate extensive Lean 4 proof data from mathematical competition problems to address the lack of training data for large language models (LLMs) in formal theorem proving.

• Automated Theorem Proving: Details how the fine-tuned DeepSeekMath model, using the synthetic dataset, achieved high whole-proof generation accuracies on the Lean 4 miniF2F test, surpassing baseline models like GPT-4, demonstrating the LLM's capability in automated theorem proving (a core formal method task).

• Formal Proof Generation: Explains that generative AI models, like LLMs, can be utilized to automatically generate and verify formal proofs in environments such as Lean 4, overcoming the challenge of manually crafting formal proofs.

Cited by 147

This relates to DeepSeek-Prover, also covered in 2024 under ***Publication 2.6*** [15]. As discussed earlier, being based on DeepSeek GenAI technology, this could be an efficient approach to theorem proving, and as such, well worth monitoring for potential formal methods use or adaptation.

Publication 2.10

DEVS Copilot: Towards Generative AI-Assisted Formal Simulation Modelling based on Large Language Models [19].

T Carreira-Munich, et al. – Proceedings, 2024 – ieeexplore.ieee.org

Explores the use of generative AI, specifically Large Language Models like GPT-4, to assist in obtaining correct executable simulation models, which are a form of formal specification.

• Uses DEVS Formalism: Adopts the Discrete Event System Specification (DEVS), a sound modeling and simulation formalism, as a suitable candidate for exploring the limits of LLM capabilities to generate correct generalized simulation models.

• Prototype Tool Development: Introduces a methodology and tool, DEVS Copilot, which transforms natural language descriptions of systems into a DEVS executable simulation model using AI, succeeding at producing correct DEVS simulations in a case study.

Cited by 5

DEVS (Discrete Event System Specification) is a modular and hierarchical formalism for modeling and analyzing systems, including discrete event systems, e.g., as described by state transition tables.

 doi:10.20944/preprints202603.0373.v1

Although DEVS was originally formulated in 1976, it does not seem to have been widely adopted, especially in the formal methods community. This 2024 conference paper presents a prototype tool, DEVS Copilot, for transforming a natural language description into a formal model, using GenAI. In practice, this is likely to need an interactive approach in general. The paper is not very well cited, so the tool looks more like a prototype example rather than one that will be adopted more generally.

Summary:

The publications returned by this research question are certainly more relevant to recent research. In particular, several are based on the Lean 4 functional language for theorem proving. This looks like an important focus for current and probably future research in this area.

Question 3: What are the latest papers investigating the use of Generative AI to support formal methods?

Finally, it was realised that Google Scholar Labs was not necessarily returning the latest publications in this fast-moving area. Earlier publications may have more citations, potentially biasing Google to return these rather then the latest research. This third research question aimed to concentrate results on the latest research papers in this area.

Publication 3.1

[PDF] arxiv.org

A new era in software security: Towards self-healing software via large language models and formal verification [20].

N Tihanyi, Y Charalambous, et al. – Proceedings, 2025 – ieeexplore.ieee.org

Presents a novel approach integrating Large Language Models (LLMs) with Formal Verification for automatic software vulnerability repair, using Bounded Model Checking (BMC) to identify vulnerabilities and extract counterexamples for the LLM.

• Introduces ESBMC-AI Framework: Introduces the ESBMC-AI framework as a proof of concept, leveraging the Efficient SMT-based Context-Bounded Model Checker (ESBMC) and a pre-trained transformer model to detect and fix errors in C programs.

• Evaluates Automatic Repair Capabilities: Evaluates the approach on 50,000 C programs and demonstrates its capability to automate the detection and repair of issues such as buffer overflow, arithmetic overflow, and pointer dereference failures with high accuracy.

Cited by 123

This 2025 conference paper presents a novel approach that is already very well cited, even though it is presented as a proof of concept. Model checking is a widely used approach for the verification of hardware and software systems with liveness and safety requirements. Bounded model checking combines model checking with satisfiability solving [39], using a SAT solver computer program to solve the Boolean satisfiability problem (SAT) to find an interpretation (assignment of a meaning) that satisfies a given Boolean formula. In this paper, SMT (Satisfiability Modulo Theories), which generalizes SAT to cover numbers, data structures, etc., is used to determine whether a mathematical formula is satisfiable. This paper specifically uses the C programming language. The approach is worth monitoring for developments in the future. See also the related ***Publication 3.3*** below.

Publication 3.2

[PDF] arxiv.org

Kimina-Prover Preview: Towards large formal reasoning models with reinforcement learning [18].

H Wang, M Unsal, X Lin, M Baksys, et al. – *arXiv* preprint, 2025 – arxiv.org

Introduces Kimina-Prover Preview, a large language model utilizing a novel reasoning-driven exploration paradigm for formal theorem proving, trained with a large-scale reinforcement learning (RL) pipeline from Qwen2.5-72B.

 doi:10.20944/preprints202603.0373.v1

• Lean 4 Proof Generation: Demonstrates strong performance in Lean 4 proof generation by employing a structured reasoning pattern, which allows the model to emulate human problem-solving strategies and iteratively refine proof steps.

• Formal Reasoning Pattern: Presents a formal reasoning pattern designed to align informal mathematical reasoning data with its translation to formal proofs, addressing a critical challenge for LLMs in this domain.

Cited by 74

This is the same paper as that discussed in ***Publication 2.8***. The Google Scholar Labs summary is slightly different in content, perhaps due to the different research question being posed. With this research question, it is rated even more highly as ***Publication 3.2***. Being based on the popular Lean 4 language and already highly-cited even though it was published in 2025, the Kimina-Prover could be an important development and worth monitoring for the future.

Publication 3.3

[PDF] acm.org

The FormAI dataset: Generative AI in software security through the lens of formal verification [21].

N Tihanyi, T Bisztray, R Jain, et al. – Proceedings, 2023 – dl.acm.org

Introduces the FormAI dataset, which consists of 112,000 AI-generated C programs classified for vulnerabilities using a formal verification method.

• Formal Verification Method: Employs the Efficient SMT-based Bounded Model Checker (ESBMC) for formal verification, which uses model checking, abstract interpretation, constraint programming, and satisfiability modulo theories to definitively detect vulnerabilities.

• AI Code Generation: Details the process of using Large Language Models (LLMs), specifically GPT-3.5-turbo, and a dynamic zero-shot prompting technique to generate diverse C programs with varying complexity.

Cited by 100

The first author is the same as for the paper returned under ***Publication 3.1*** [20]. This paper is an earlier 2023 conference paper. Although this paper is well cited, most likely, ***Publication 3.1*** is best for monitoring the latest research in this area.

Publication 3.4

[PDF] acm.org

Proof automation with large language models [22].

M Lu, B Delaware, T Zhang – Proceedings, 2024 – dl.acm.org

Proposes PALM, a novel proof automation approach that combines Large Language Models (LLMs) and symbolic methods in a generate-then-repair pipeline to create formal proofs.

• Evaluates PALM Performance: Evaluates the PALM system on a large dataset of over 10,000 theorems, demonstrating significant improvement over existing state-of-the-art methods.

• Identifies LLM Errors: Identifies common errors made by LLMs, such as GPT-3.5, when generating formal proofs, noting that they often succeed with high-level structure but fail with low-level details.

Cited by 36

This 2024 conference paper proposes a new proof automation approach, dubbed PALM, to use LLMs in the generation of formal proofs for theorems, using the Coq theorem prover (renamed "Rocq" in 2025, https://rocq-prover.org) and GPT-3.5. It notes that GPT-3.5 can generate a valid high-level structure for proofs, but makes mistakes at lower levels. Thus, the approach allows iterative correction of the lower levels. The paper claims to outperform other state-of-the-art approaches to proof generation. The approach looks promising, although the number of citations could be higher. This may be due to

 doi:10.20944/preprints202603.0373.v1

the use of Coq rather than Lean; the latter seems to have a greater research focus at present. Note that the paper is also available on the *arXiv* archive (https://doi.org/10.48550/arXiv.2409.14274), no doubt for speed of dissemination and easy open access.

Publication 3.5

[PDF] neurips.cc

Autoformalization with large language models [23].

Y Wu, AQ Jiang, W Li, et al. – Proceedings, 2022 – proceedings.neurips.cc

Explores the use of Large Language Models (LLMs) for autoformalization, the process of translating natural language mathematics into formal specifications and proofs.

• Translating Math Problems: Shows that LLMs can translate a significant portion of mathematical competition problems perfectly to formal specifications in Isabelle/HOL, indicating their capability in generating formal language from natural language.

• Improving Theorem Prover: Demonstrates the usefulness of autoformalization by improving a neural theorem prover, resulting in a new state-of-the-art result on the MiniF2F theorem proving benchmark.

Cited by 311

This 2022 conference paper is highly cited. It looks at transforming natural language mathematical descriptions into complete formal specifications with associated proofs. The paper uses the well-established Isabelle/HOL theorem prover [54]. Related developments include the DeepIsaHOL project (https://github.com/yonoteam/DeepIsaHOL), using reinforcement learning (RL), machine learning paradigm where an autonomous agent learns to make decisions by trial and error, to improve proof-automation in theorem proving using the Isabelle proof assistant.

Publication 3.6

[PDF] hal.science

Applications of AI to study of finite algebraic structures and automated theorem proving [24].

B Shminke – Thesis, 2023 – theses.hal.science

Proposes a generative neural network architecture for creating finite models of algebraic structures, drawing inspiration from image generation models like Generative Adversarial Networks (GANs) and autoencoders.

• Semigroup Generation Package: Contributes a Python package for generating small-sized finite semigroups, serving as a reference implementation of the proposed generative method.

• Reinforcement Learning for Provers: Designs a general architecture for guiding saturation provers using reinforcement learning algorithms, which is a key component of automated theorem proving (a formal method).

Cited by 0

Unusually, compared to most of the publications included, this is a PhD research thesis rather than a journal or conference paper. It dates from 2023, which, in this fast-moving area, may already be dated. It has no citations on Google Scholar, which also raises questions over its inclusion and influence. As introduced under ***Publication 3.5***, it uses reinforcement learning. Without associated peer-reviewed papers, this research may have little impact in practice.

Publication 3.7

[PDF] arxiv.org

Specify what? Enhancing neural specification synthesis by symbolic methods [25].

G Granberry, W Ahrendt, M Johansson – Proceedings, 2024 – Springer

 doi:10.20944/preprints202603.0373.v1

Investigates how combinations of Large Language Models (LLMs) and symbolic analyses can synthesize specifications for C programs in the ACSL language.

• Augmenting LLMs with Formal Tools: Augments LLM prompts with outputs from formal methods tools, specifically Pathcrawler and EVA from the Frama-C ecosystem, to produce C program annotations.

• Impact of Symbolic Analysis: Demonstrates that integrating symbolic analysis enhances the quality of annotations, making them more context-aware and attuned to runtime errors, and enabling inference of program intent rather than just behavior.

Cited by 11

This 2024 conference paper is not very well cited. The ACSL language is the ANSI/ISO C Specification Language, a behavioral specification language for C programs. The paper investigates the use of LLMs in synthesizing C program specifications in ACSL. It uses tools from Frama-C (https://frama-c.com), an open-source, extensible, and collaborative platform for source-code analysis of C programs. Given the low number of citations, it is not clear if this approach will be widely adopted in practice.

Publication 3.8

[PDF] arxiv.org

Generative AI augmented induction-based formal verification [26].

A Kumar, DN Gadde – Proceedings, 2024 – ieeexplore.ieee.org

Demonstrates how Generative AI can be used in induction-based formal verification to increase the verification throughput.

• Automating Assertion Generation: Explores the potential of GenAI to automate assertion generation, which helps resolve the complexity and labor-intensive nature of formal verification.

• GenAI-Based Verification Flow: Presents a GenAI-based flow that generates helper assertions and generates lemmas in the event of an inductive step failure during formal verification.

Cited by 7

This 2024 conference paper claims to demonstrate the use of GenAI to improve the efficiency of formal verification. The subject area certainly looks very relevant. However, given the low number of citations, its influence may have a low impact in practice.

Publication 3.9

[PDF] arxiv.org

Autoformalization in the Era of Large Language Models: A Survey [27].

K Weng, L Du, S Li, W Lu, H Sun, et al. – *arXiv* preprint, 2025 – arxiv.org

Examines how autoformalization, driven by Large Language Models (LLMs), is applied across various mathematical domains and levels of difficulty, relating to automated theorem proving.

• Autoformalization Workflow Analysis: Analyzes the end-to-end workflow of autoformalization, including data preprocessing, model design, and evaluation protocols for converting informal math to formal representations.

• Verifiability of LLM Outputs: Explores the emerging role of autoformalization in enhancing the verifiability and trustworthiness of LLM-generated outputs, bridging the gap between human intuition and machine-verifiable reasoning.

Cited by 11

 doi:10.20944/preprints202603.0373.v1

This is a 2025 survey paper, which can be a useful starting point for investigating further research. Survey papers can receive a large number of citations if they are influential, but this has not yet happened with this paper.

Publication 3.10

[PDF] arxiv.org

Vulnerability detection: from formal verification to large language models and hybrid approaches: a comprehensive overview [28].

N Tihanyi, T Bisztray, MA Ferrag, et al. – *arXiv* preprint, 2025 – arxiv.org

> Presents a comprehensive study of state-of-the-art software testing and verification, focusing on classical formal methods, Large Language Model (LLM)-based analysis, and emerging hybrid techniques.
>
> • Enhancing Verification Scalability: Highlights the potential of hybrid systems, which combine formal rigor with LLM-driven insights, to enhance the effectiveness and scalability of software verification.
>
> • LLM-Assisted Formal Verification: Details how integrating LLMs with formal methods and program analysis tools is a promising direction for improving software vulnerability detection by reducing the manual burden through automatic generation of formal assertions/invariants.
>
> Cited by 6

The first author is the same as for the paper returned under ***Publication 3.1*** [20] and the first two authors are the same as the paper under ***Publication 3.3*** [21]. Like ***Publication 3.9***, this is a survey paper on the *arXiv* archive, but despite this, it has not yet been widely cited. Since the Google Scholar Labs search, the paper has been formally published in 2026 [29] so at least it has now been peer reviewed.

Summary:

The publications for this research question are at least all recent (one from 2022, two from 2023, three from 2024, and four from 2025). An issue is that some are not well cited. On the other hand, some of these recent papers are very well cited already. It is these that are already influential and may lead to further worthwhile developments.

4. Discussion

4.1. The GenAI Survey

The use of Google Scholar Labs to answer the research questions in Section 3 provides some useful results, but these still need to be interpreted by an expert in the field. The main benefit is the speeding up of the process to survey existing literature. However, some results may be less relevant, as determined by detailed inspection of the results. Research questions can be refined very quickly by assessing the results and then making a more precise enquiry depending on the issues found in the returned publications. In this paper, this has been done with three research questions, but the process could easily be continued at speed and interactively until enough relevant results are obtained.

A notable feature of the publications in the survey is the number of papers available through the *arXiv* online archive. This allows rapid publication by researchers, with checking for appropriateness, but not formal peer reviewing. The developments in AI are so fast that authors appear to prefer quick publication of their latest research. Even many of the papers that have been formally published also have versions that are freely available on *arXiv*. And, of course, articles on *arXiv* can be peer-reviewed and published subsequently (e.g., as is the case for ***Publication 3.10***).

Although there is some overlap of results returned by the three research questions, this is remarkably low, and has been commented on in the results where it did occur. Whether this low overlap is

 doi:10.20944/preprints202603.0373.v1

a good or bad aspect is difficult to determine, but perhaps makes the results from multiple related research questions more useful by avoiding repetition.

A useful feature of Google Scholar is the number of citations by other publications in the Google Scholar database. This helps to indicate the impact of a publication, bearing in mind the number of years since it appeared. This has been commented on for many of the answers in the results. Even some of the recent publications are already highly cited, indicating their rapid influence. Older papers, even if well cited, are less likely to be relevant for the research area under investigation.

Google Scholar has the ability generate BIBTEX-format records for citations, convenient for use with the LATEX document preparation system, widely used by scientists in writing papers, including the present paper. However, the BIBTEX records are not very complete, missing out book series information, publisher locations, etc. Most importantly, DOIs (Digital Object Identifiers) are not included for publications. These provide reliable and immutable online access to publications, unlike URLs (Uniform Resource Locators), which can vary at any time and may be short-lived. Therefore, although this paper used the Google Scholar generation of BIBTEX records for the surveyed publications initially, significant manual effort was needed to add further information, especially DOIs where available.

4.2. Future Directions

As we look to the future (2026 and beyond), the landscape of formal methods have the potential to converge with the recent rapid advancements in Artificial Intelligence (AI). Traditionally, there has been a dichotomy:

- **Formal Methods:** Rigorous, "correct by construction", based on precise language and proof assistants.
- **Machine Learning/AI:** Based on large learned models, statistical probability, and often "unreliable" or "hallucinating" outputs.

However, we are seeing the emergence of hybrid approaches. Just as deep neural networks have revolutionized pattern recognition, AI is now being applied to mathematical discovery and program proof [31,60].

Recent research highlights the potential for AI to guide search in theorem proving. For example, systems like *Lean* (a proof assistant and programming language) are being integrated with AI tools to suggest outline proofs, which are then rigourously checked by the formal system [51,52]. This addresses the "unreliability" of LLMs by using the formal system as a verifier of the AI's "intuition". New concepts have emerged and are likely to develop further:

- **Vibe Coding:** A colloquial term for AI-generated code that "feels" right but requires verification.
- **Vibe Proving:** Perhaps using AI to bridge gaps in formal proofs.

Publications in *Nature* regarding mathematical discoveries from program search [55] and the ideas of the Fields Medalist Terence Tao [60] suggest we are entering a new and exciting era where AI fills in the steps of a proof, easing the use of formal verification in an interactive approach to proving both mathematics in general and programs in particular.

Just as AI experienced "winters" before its current boom, formal methods have seen cycles of hype and disillusionment. The integration with GenAI may trigger a "Formal Methods Spring". Tools like *DeepSeekMath* [58] and *LeanProgress* [51] indicate that the barrier to entry for formal proofs is lowering. We may soon see a unification where engineers calculate and prove properties of software with the same ease that civil engineers calculate loads, aided by intelligent assistants.

5. Conclusions

Formal methods have traveled a long path from Turing's manual checking of routines to industrial-scale use, including semi and even fully automated verification. The author believes that the future lies in the symbiotic relationship between formal rigor and artificial intelligence. By leveraging AI

 doi:10.20944/preprints202603.0373.v1

to handle the tedious aspects of proof and specification, while retaining the mathematical certainty of formal methods for verification, it should be possible to achieve a level of software reliability previously thought impossible. As Théophile Gautier (1811–1872) wrote on art in 1856 [43]:

> Oui, l'ouvre sort plus belle
> D'une forme au travail
> Rebelle,
> Vers, marbre, onyx, émail.
>
> [Yes, the work comes out more beautiful from a material that resists the process, verse, marble, onyx, or enamel.]

In the context of this paper, software is that resistant material, and formal methods, potentially augmented by AI, provide the chisel and mallet to shape it.

References

1. Ramsay, A. *Formal Methods in Artificial Intelligence*; Vol. 6, Cambridge University Press, 1988.
2. Krichen, M.; Mihoub, A.; Alzahrani, M.Y.; Adoni, W.Y.H.; Nahhal, T. Are formal methods applicable to machine learning and Artificial Intelligence? In Proceedings of the 2022 2nd International Conference of Smart Systems and Emerging Technologies (SMARTTECH). IEEE, 2022, pp. 48–53. https://doi.org/10.1109/SMARTTECH54121.2022.00025.
3. Szegedy, C. A promising path towards autoformalization and General Artificial Intelligence. In Proceedings of the International Conference on Intelligent Computer Mathematics, Cham, 2020; Vol. 12236, *LNCS*, pp. 3–20. https://doi.org/10.1007/978-3-030-53518-6_1.
4. Wang, H. Computer theorem proving and Artificial Intelligence. In *Computation, Logic, Philosophy: A Collection of Essays*; Springer: Dordrecht, 1990; Vol. 2, *Mathematics and its Application (China Series)*, pp. 63–75. https://doi.org/10.1007/978-94-009-2356-0_5.
5. Cai, C.H.; Sun, J.; Dobbie, G. Automatic B-model repair using model checking and machine learning. *Automated Software Engineering* **2019**, *26*, 653–704. https://doi.org/10.1007/s10515-019-00264-4.
6. Sun, X.; Khedr, H.; Shoukry, Y. Formal verification of neural network controlled autonomous systems. In Proceedings of the Proceedings of the 22nd ACM International Conference on Hybrid Systems: Computation and Control. ACM, 2019, pp. 147–156. https://doi.org/10.1145/3302504.331180.
7. Zhang, Y.; Cai, Y.; Zuo, X.; Luan, X.; Wang, K.; Hou, Z.; Zhang, Y.; Wei, Z.; Sun, M.; Sun, J.; et al. The fusion of Large Language Models and formal methods for trustworthy AI agents: A roadmap. *arXiv preprint* **2024**, *arXiv:2412.06512*. https://doi.org/10.48550/arXiv.2412.06512.
8. Bauer, A.; Petković, M.; Todorovski, L. MLFMF: Data sets for machine learning for mathematical formalization. *Advances in Neural Information Processing Systems (NeurIPS 2023)* **2023**, *36*, 50730–50741. https://doi.org/10.48550/arXiv.2310.16005.
9. Bortolussi, L.; Milios, D.; Sanguinetti, G. Machine learning methods in statistical model checking and system design – tutorial. In Proceedings of the Runtime Verification; Bartocci, E.; Majumdar, R., Eds., Cham, 2015; Vol. 9333, *LNCS*, pp. 323–341. https://doi.org/10.1007/978-3-319-23820-3_23.
10. Kuznetsov, S.O. Machine learning and formal concept analysis. In Proceedings of the International Conference on Formal Concept Analysis; Eklund, P., Ed., Cham, 2004; Vol. 2961, *LNCS*, pp. 287–312. https://doi.org/10.1007/978-3-540-24651-0_25.
11. Jha, S.; Jha, S.K.; Lincoln, P.; Bastian, N.D.; Velasquez, A.; Neema, S. Dehallucinating Large Language Models using formal methods guided iterative prompting. In Proceedings of the 2023 IEEE International Conference on Assured Autonomy (ICAA). IEEE, 2023, pp. 149–152. https://doi.org/10.1109/ICAA58325.2023.00029.
12. Polu, S.; Sutskever, I. Generative language modeling for automated theorem proving. *arXiv preprint* **2020**, *arXiv:2009.03393*. https://doi.org/10.48550/arXiv.2009.03393.
13. Lin, Y.; Tang, S.; Lyu, B.; Wu, J.; Lin, H.; Yang, K.; Li, J.; Xia, M.; Chen, D.; Arora, S.; et al. Goedel-Prover: A frontier model for open-source automated theorem proving. *arXiv preprint* **2025**, *arXiv:2502.07640*. https://doi.org/10.48550/arXiv.2412.06512.
14. Lin, Y.; Tang, S.; Lyu, B.; Yang, Z.; Chung, J.H.; Zhao, H.; Jiang, L.; Geng, Y.; Ge, J.; Sun, J.; et al. Goedel-Prover-V2: Scaling formal theorem proving with scaffolded data synthesis and self-correction. *arXiv preprint* **2025**, *arXiv:2508.03613*. https://doi.org/10.48550/arXiv.2508.03613.

 doi:10.20944/preprints202603.0373.v1

15. Xin, H.; Ren, Z.; Song, J.; Shao, Z.; Zhao, W.; Wang, H.; Liu, B.; Zhang, L.; Lu, X.; Du, Q.; et al. DeepSeek-Prover-V1.5: Harnessing proof assistant feedback for reinforcement learning and Monte-Carlo tree search. *arXiv preprint* **2024**, *arXiv:2408.08152*. https://doi.org/10.48550/arXiv.2408.08152.
16. Xin, H.; Guo, D.; Shao, Z.; Ren, Z.; Zhu, Q.; Liu, B.; Ruan, C.; Li, W.; Liang, X. DeepSeek-Prover: Advancing theorem proving in LLMs through large-scale synthetic data. *arXiv preprint* **2024**, *arXiv:2405.14333*. https://doi.org/10.48550/arXiv.2405.14333.
17. Katz, S.M.; Corso, A.L.; Strong, C.A.; Kochenderfer, M.J. Verification of image-based neural network controllers using generative models. *Journal of Aerospace Information Systems* **2022**, *19*, 574–584. https://doi.org/10.2514/1.I011071.
18. Wang, H.; Unsal, M.; Lin, X.; Baksys, M.; Liu, J.; Santos, M.D.; Sung, F.; Vinyes, M.; Ying, Z.; Zhu, Z.; et al. Kimina-Prover preview: Towards large formal reasoning models with reinforcement learning. *arXiv preprint* **2025**, *arXiv:2504.11354*. https://doi.org/10.48550/arXiv.2504.11354.
19. Carreira-Munich, T.; Paz-Marcolla, V.; Castro, R. DEVS Copilot: Towards Generative AI-Assisted Formal Simulation Modelling based on Large Language Models. In Proceedings of the 2024 Winter Simulation Conference (WSC). IEEE, 2024, pp. 2785–2796. https://doi.org/10.1109/WSC63780.2024.10838994.
20. Tihanyi, N.; Charalambous, Y.; Jain, R.; Ferrag, M.A.; Cordeiro, L.C. A new era in software security: Towards self-healing software via Large Language Models and formal verification. In Proceedings of the 2025 IEEE/ACM International Conference on Automation of Software Test (AST). IEEE, 2025, pp. 136–147. https://doi.org/10.1109/AST66626.2025.00020.
21. Tihanyi, N.; Bisztray, T.; Jain, R.; Ferrag, M.A.; Cordeiro, L.C.; Mavroeidis, V. The FormAI dataset: Generative AI in software security through the lens of formal verification. In Proceedings of the PROMISE 2023: Proceedings of the 19th International Conference on Predictive Models and Data Analytics in Software Engineering. ACM, 2023, pp. 33–43. https://doi.org/10.1145/3617555.361787.
22. Lu, M.; Delaware, B.; Zhang, T. Proof automation with Large Language Models. In Proceedings of the ASE'24: Proceedings of the 39th IEEE/ACM International Conference on Automated Software Engineering. IEEE/ACM, 2024, pp. 1509–1520. https://doi.org/10.1145/3691620.3695521.
23. Wu, Y.; Jiang, A.Q.; Li, W.; Rabe, M.; Staats, C.; Jamnik, M.; Szegedy, C. Autoformalization with Large Language Models. In Proceedings of the NIPS'22: Proceedings of the 36th International Conference on Advances in Neural Information Processing Systems; Koyejo, S.; Mohamed, S.; Agarwal, A.; Belgrave, D.; Cho, K.; Oh, A., Eds. Curran Associates, 2022, Vol. 35, pp. 32353–32368. https://doi.org/10.5555/3600270.3602614.
24. Shminke, B. Applications of AI to study of finite algebraic structures and automated theorem proving. PhD thesis, Université Côte d'Azur, 2023.
25. Granberry, G.; Ahrendt, W.; Johansson, M. Specify what? Enhancing neural specification synthesis by symbolic methods. In Proceedings of the Integrated Formal Methods (IFM 2024); Kosmatov, N.; Kovács, L., Eds., Cham, 2024; Vol. 15234, *LNCS*, pp. 307–325. https://doi.org/10.1007/978-3-031-76554-4_19.
26. Kumar, A.; Gadde, D.N. Generative AI augmented induction-based formal verification. In Proceedings of the 2024 IEEE 37th International System-on-Chip Conference (SOCC). IEEE, 2024, pp. 1–2. https://doi.org/10.1109/SOCC62300.2024.10737803.
27. Weng, K.; Du, L.; Li, S.; Lu, W.; Sun, H.; Liu, H.; Zhang, T. Autoformalization in the Era of Large Language Models: A Survey. *arXiv preprint* **2025**, *arXiv:2505.23486*. https://doi.org/10.48550/arXiv.2505.23486.
28. Tihanyi, N.; Bisztray, T.; Ferrag, M.A.; Cherif, B.; Dubniczky, R.A.; Jain, R.; Cordeiro, L.C. Vulnerability detection: From formal verification to Large Language Models and hybrid approaches: A comprehensive overview. *arXiv preprint* **2025**, *arXiv:2503.10784*. https://doi.org/10.48550/arXiv.2503.10784.
29. Tihanyi, N.; Bisztray, T.; Ferrag, M.A.; Cherif, B.; Dubniczky, R.A.; Jain, R.; Cordeiro, L.C. Vulnerability detection: From formal verification to Large Language Models and hybrid approaches: A comprehensive overview. In *Adversarial Example Detection and Mitigation Using Machine Learning*; Nowroozi, E.; Taheri, R.; Cordeiro, L., Eds.; Springer: Cham, 2026; chapter 3, pp. 33–47. https://doi.org/10.1007/978-3-031-99447-0_3.
30. Abrial, J.-R. *The B-Book: Assigning programs to meanings*. Cambridge University Press (1996).
31. Avigad, J. Mathematics in the Age of AI. LMS/BCS-FACS Seminar, London Mathematical Society (6 November 2025). https://www.lms.ac.uk/events/lms-bcs-facs-seminar-jeremy-avigad
32. Beth, E.W. *Formal Methods: An Introduction to Symbolic Logic and to the Study of Effective Operations in Arithmetic and Logic*. D. Reidel Publishing Company / Dordrecht-Holland (1962).
33. Boulanger, J.-L. (ed.) *Formal Methods: Industrial Use from Model to the Code*. ISTE, Wiley (2012).

 doi:10.20944/preprints202603.0373.v1

34. Bowen, J.P., The Z Notation: Whence the Cause and Whither the Course? In Liu, Z., Zhang, Z. (eds.) *Engineering Trustworthy Software Systems, SETSS 2014*. LNCS, vol. **9506**, pp. 103–151. Cham: Springer (2016). https://doi.org/10.1007/978-3-319-29628-9_3
35. Bowen, J.P., Hinchey, M.G. Ten commandments of formal methods. *IEEE Computer* **28**(4), 56–63 (1995). https://doi.org/10.1109/2.375178
36. Bowen, J.P., Hinchey, M.G. Seven more myths of formal methods. *IEEE Software* **12**(4), 34–41 (1995). https://doi.org/10.1109/52.391826
37. Bowen, J.P., Stavridou, V. Safety-critical systems, formal methods and standards. *Software Engineering Journal* **8**(4), 189–209 (1993). https://doi.org/10.1049/sej.1993.0025
38. Brereton, P., Kitchenham, B.A., Budgen, D., Turner, M., Khalil, M. Lessons from applying the systematic literature review process within the software engineering domain. *The Journal of Systems and Software* **80**(4), 571–583 (2007). https://doi.org/10.1016/j.jss.2006.07.009
39. Clarke, E., Biere, A., Raimi, R., Zhu, Y. Bounded Model Checking Using Satisfiability Solving. *Formal Methods in System Design* **19**, 7–34 (2001). https://doi.org/10.1023/A:1011276507260
40. Dieste, O., Grimán, A., Juristo, N. Developing search strategies for detecting relevant experiments. *Empirical Software Engineering* **14**, 513–539 (2009). https://doi.org/10.1007/s10664-008-9091-7
41. Floyd, R.W. Assigning meaning to programs. In: *Proceedings of Symposia in Applied Mathematics* **19**. Also in: Scharts, S.T. (ed.), *Mathematical Aspects of Computer Science*. Providence: American Mathematical Society (1967). https://doi.org/10.1007/978-94-011-1793-7_4 (republished in 1993).
42. Frappier, M., Habrias, H. (eds.) *Software Specification Methods: An Overview Using a Case Study*. Springer, FACIT (2001).
43. Gautier, T. *L'Art Moderne*. Paris: M. Lévy Frères (1856). https://archive.org/details/lartmodernetho00gautuoft
44. Gnesi, S., Margaria, T. *Formal Methods for Industrial Critical Systems: A Survey of Applications*. IEEE Computer Society Press, Wiley (2012).
45. Google. Google Scholar Labs. https://scholar.google.com/scholar_labs/search (accessed 25 February 2026).
46. Hall, J.A. Seven myths of formal methods. *IEEE Software* **7**(5), 11–19 (1990). https://doi.org/10.1109/52.57887
47. Hinchey, M.G., Bowen, J.P. (eds.) *Applications of Formal Methods*. Prentice Hall, Series in Computer Science (1995).
48. Hinchey, M.G., Bowen, J.P. (eds.) *Industrial-Strength Formal Methods in Practice*. Springer, FACIT (1999).
49. Hoare, C.A.R. An axiomatic basis for computer programming. *Communications of the ACM* **12**(10), 576–580 (1969). https://doi.org/10.1145/363235.363259
50. Hoare, C.A.R., He, J. *Unifying Theories of Programming*. Prentice Hall, Series in Computer Science (1998).
51. Huang, S., et al. LeanProgress: Guiding Search for Neural Theorem Proving via Proof Progress Prediction. *arXiv* (February 2025). https://doi.org/10.48550/arXiv.2502.17925
52. Lu, J., et al. Lean Finder: Semantic Search for Mathlib That Understands User Intents. *arXiv* (October 2025). https://doi.org/10.48550/arXiv.2510.15940
53. Morris, F.L., Jones, C.B. An Early Program Proof by Alan Turing. *IEEE Annals of the History of Computing* **6**(2), 139–143 (1984). https://doi.org/10.1109/MAHC.1984.10017
54. Nipkow, T., Wenzel, M., Paulson, L.C. (eds.) *Isabelle/HOL: A Proof Assistant for Higher-Order Logic*. LNCS, vol. **2283**. Cham: Springer (2002). https://doi.org/10.1007/3-540-45949-9
55. Romera-Paredes, B., et al. Mathematical discoveries from program search and large language models. *Nature* **625**, 468–475 (2024). https://doi.org/10.1038/s41586-023-06924-6
56. Petersen, K., Feldt, R., Mujtaba, S., Mattsson, M. Systematic mapping studies in software engineering In *EASE'08: Proceedings of the 12th International Conference on Evaluation and Assessment in Software Engineering*, pp. 68–77 (2008).
57. Petersen, K., Vakkalanka, S., Kuzniarz, L. Guidelines for conducting systematic mapping studies in software. *Information and Software Technology* **64**, 1–18 (2015). https://doi.org/10.1016/j.infsof.2015.03.007
58. Shao, Z., et al. DeepSeekMath-V2: Towards Self-Verifiable Mathematical Reasoning. *arXiv* (November 2025). https://doi.org/10.48550/arXiv.2511.22570
59. Strachey, C. The Strachey's Philosophy. Department of Computer Science, University of Oxford. https://www.cs.ox.ac.uk/activities/concurrency/courses/stracheys.html (accessed 2 March 2025).
60. Tao, T. The Potential for AI in Science and Mathematics. Oxford Mathematics Public Lectures, Science Museum, London (17 July 2024). https://www.maths.ox.ac.uk/node/68243

 doi:10.20944/preprints202603.0373.v1

61. Turing, A.M. Checking a Large Routine. In: *Report of a Conference on High Speed Automatic Calculating Machines*, Cambridge University Mathematical Laboratory, pp. 67–69 (1949). https://turingarchive.kings.cam.ac.uk/checking-large-routine
62. Turing, A.M. Computing Machinery and Intelligence, *Mind* **LIX**(236), 433–460 (1952). https://doi.org/10.1093/mind/LIX.236.433
63. Whitehead A.N., Russell, B. *Principia Mathematica*. Cambridge University Press (1910–1913).
64. Woodcock, J., Larsen, P.G., Bicarregui, J., Fitzgerald, J. Formal methods: Practice and experience *ACM Computing Surveys (CSUR)*, **41**(4), 1–36 (2009). https://doi.org/10.1145/1592434.1592436