1 Introduction

Proof assistant systems have been used to formalize and prove a wide range of interesting mathematical theorems and results. Well-known achievements including the use of the LEAN proof system [27] to prove the Polynomial Freiman-Ruzsa Conjecture [93] by Terence Tao, the theorem by Peter Scholze used in the Liquid tensor experiment [19], and many more as shown in [47]. However, writing proofs in proof assistants such as LEAN can be challenging, especially for non-experts, as it requires a deep understanding of the proof assistant's language and the ability to construct effective proofs using a series of tactics.

In recent years, the advent of practical large language models (LLMs) in AI/ML has led to significant integration and advancements in formal method (FM) techniques, e.g., program synthesis and verification synthesis [20, 21, 50, 82]. Unsurprisingly, researcher has leveraged LLMs to enhance the capabilities of proof assistants, e.g., by automating proof generation [30, 44]. For example, various work [101,106] has integrated AI, e.g., as "copilot", to assist mathematicians in automatically generating proof steps that can be checked with LEAN.

Despite these advancements, several limitations exist in the current state of AI-assisted theorem proving. First, formalizing and proving mathematical theorems in LEAN remains difficult, especially in areas such as combinatorics or graphs theory where more complex combinatorial proofs are often used and existing LLMs lack of training data in such domains. Second, the current AI search capabilities within proof assistants are often inadequate, returning irrelevant results that do not help LEAN to make progress in proving theorems. Third, proof assistant systems such as LEAN are often provide errors/feedback which are often cryptic and can misguide users into finding wrong tactics and theorems. Even existing theorems and tactics in LEAN are often difficult to find, making it hard for users to find the appropriate tactics and theorems and in many cases rewrite the same tactics and theorems.

1.1 Proposed Work

This proposal aims to develop Mathscape, an AI-enhanced theorem prover that leverages the latest advancements in AI/ML and uses the LEAN theorem prover for the purpose of automatic proof generation and verification of mathematical theorems. The proposed research focuses on three key components: expanding mathematical reasoning in new domains, advancing AI/ML methodologies to aid proof generation, and enhancing the usability of the LEAN proof assistant. The project will explore the following research directions:

Mathematical Reasoning Enhancement: We will expand AI-assisted mathematical reasonings in *combinatorics* and *graph theory*, which are important in maths, CS, and many other fields, but often require complex proofs that are difficult for AI to generate. We will focus on identifying and formalizing combinatorial identities and proofs, dissecting complex combinatorial identities into simpler components, and exploring bijective proofs.

Advancing AI/ML Methodologies: To improve AI's search for relevant tactics and theorems, we will develop a new AI model that can generate tactics and theorems based on the current proof state. This includes designing a "Chain of Thought" methodology to generate better, more relevant LEAN proofs. We will also a new ML dataset, customized for the considered domains and use active learning techniques to annotate the most informative questions. Lastly, we will develop a customized LLM, leveraging meta-learning approaches to adapt effectively using limited data.

Improving LEAN: We will make LEAN more accessible and effective. This includes LeanMatBridge, which integrates LEAN with MATLAB, allowing users to leverage MATLAB's computational capabilities within Lean's formal verification framework. Additionally, the project will develop FeedBackAnalyzer to improve the interpretability of Lean's feedback and SmartSearch, an advanced search engine to enhance the discoverability of Lean's tactics and theorems.

This interdisciplinary research combines the strengths of formal methods, AI/ML, and mathematical reasoning to develop automated and power tools and methodologies that will significantly enhance mathematical reasoning and theorem proving in LEAN and beyond. Additionally, the project will integrate these advancements into educational and outreach initiatives, fostering a deeper understanding of AI-assisted formal methods among students, mathematicians, and researchers.

1.2 PI's Qualifications and Preparations

About the PIs This project involves PIs with expertise in mathematics, formal methods (FM), and AI/ML at George Mason (Mason) University. PI. **Agnarsson** works in pure mathematics and theoretical computer science. His research is mainly in (i) graph theory, in particular in graph coloring, extremal graph theory and graph algorithms [1, 8], (ii) combinatorics and partially ordered sets [3], (iii) commutative and non-commutative ring theory, algebras over fields [7] and (iv) theoretical computer science [6].

PI. **Liu** has expertise in AI/ML, focusing on statistical learning theory, deep learning and representation learning. He regularly publishes papers in leading AI/ML conferences including NeurIPS [14, 23, 24, 33, 37, 42, 48, 54, 57, 59, 61–63, 104], ICML [13, 34, 60], ICLR [22, 36, 55, 58]. His work on continual learning [33], federated learning [63], bilevel optimization and representation learning [36, 42] have been selected as spotlight presentations at NeurIPS and ICLR (5% acceptance rate). The PI's work on topics relevant to this proposal includes nonconvex optimization [36, 52, 55, 58, 59, 87], large-scale distributed learning [22, 23, 25, 34, 59], continual learning [33, 37, 111], online learning [60, 62], convex optimization [49, 56, 57, 104], and statistical learning theory [13, 48, 61] and deep learning theory [14]. He was recognized in the *AAAI 2024 New Faculty Highlight program*.

PI. **Nguyen** works in software engineering and formal methods, with recent focus on deep neural networks (DNN) verification [28, 80]. His NeuralSAT DNN verifier [86] is ranked 4th overall and won the *"New Comer Award"* at VNN-COMP'23 [18]. Nguyen also works on invariant generation [40, 46, 68, 72, 74, 78], automatic program repair, program analyses (verification, synthesis, and repair) [45, 64, 79, 100, 108, 109]. His dynamic invariant generation work DIG [70] was recognized with the Distinguished Paper award at ICSE'14 [73], and his program repair work GenProg with 10-year Impact Paper Award at GECCO'19 [31] and 10-year Most Influential Paper Award at ICSE'19 [100].

This is the first time that the faculty at the CS and Mathematics departments at Mason collaborate. In addition to research advancements, we believe that this project will help foster interdisciplinary research and collaboration among the two departments.

Preparations To prepare for this project, the PIs and their students have developed a Mathscape prototype (§ 4.1) that integrates LEAN with GPT LLM models to prove theorems and conjectures in several domains, especially combinatorics and graph theory. This prototype will be used as a foundation for the proposed research. Our experiences in releasing benchmark and research tools (e.g., the NeuralSAT DNN verification tools from PI Nguyen) and contributing to large open-source projects (e.g., PI Nguyen wrote the initial Python API helper [94] for Z3 [65], created the comprehensive nonlinear benchmark suite [95] for the annual Software Verification competitions [17],

and participate in VNN-COMP), will help with the management and distribution of tools and datasets.

2 Broader Impacts

Our project *benefits society* by leveraging the increasingly deepening interaction of AI and formal methods to enhance mathematical reasoning. Note that while we focus on mathematical reasoning, the techniques developed can be applied to a wide range of domains and aid researchers and developers involved in high-assurance systems that might leverage formal proofs, e.g., autonomous driving, aircraft collision control, and financial modeling.

Our work will *contribute to AI/ML, formal methods, and mathematical reasoning and proofs.* By incorporating AI techniques into proof systems such as Lean, we expect to broaden the accessibility of formal methods, allowing researchers not familiar with mechanical theorem proving systems, such as mathematicians and theoretical physics, to verify their conjectures with greater ease. We will share our advancements with both the AI/ML, formal methods, and mathematics communities.

The outcomes of this project will also be integrated into the PIs' academic courses and extracurricular activities. We plan to develop an online course module on "Formal Methods and Mathematical Reasoning in the age of AI" and a series of interactive tutorials demonstrating the application of AI techniques to mathematical proofs. Our track record of involving undergraduate students in research will be continued and expanded through this project, providing them with first-hand experience in cutting-edge interdisciplinary research among FM, AI, and Maths. Furthermore, we aim to extend our outreach to K-12 students, introducing them to the basics of AI and formal reasoning, as outlined in §7.2.

3 Background

Mathematical Proof To establish the truth of mathematical statements, human often rely on intuitive reasoning and natural language explanations. However, such *informal reasoning* can be ambiguous, incomplete, or difficulty to verify, as they often rely on human intuition and interpretation, leading to potential errors and inconsistencies.

To address these issues, researchers have developed proof systems that formalize and verify mathematical proofs. This field of *formal theorem proving* can be broadly classified as *automated theorem proving* (ATP) and *interactive theorem proving* (ITP). ATP aims to verify formal statements automatically. Well-known ATP examples include SMT solvers that automatically proves theorems in first-order logic, e.g., Z3 [26] or CVC5 SMT solvers, are industrial-strength tools used for software verification and synthesis. In contrast, ITP requires human-guided proof construction using a *proof assistant*, which provides a higher-order and more expressive language for expression mathematical statements and proofs. Popular ITP proof assistants include LEAN [27, 66], Coq [15], and Isabelle [81].

LEAN LEAN is a modern and powerful ITP proof assistant that supports formal reasoning and verification of mathematical statements. To prove a theorem—an initial goal—in Lean, the user formalizes the theorem statement and then provides its proofs in Lean's language. The proof is a series of *tactics* to either finish the goal or decompose it into subgoals. The user constructs and applies tactics to each subgoal until all subgoals are proven, thereby completing the proof.

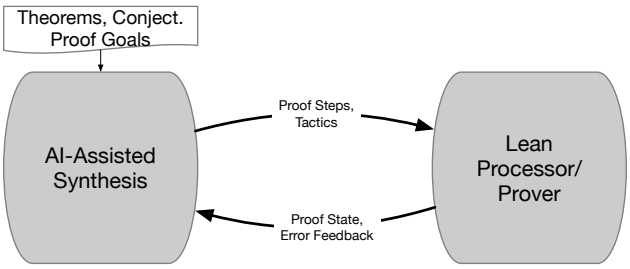

Fig. 1: Iterative Guess-and-Check Approach of Mathscape.

While allowing for proving complex theorems, Lean, being an ITP system, requires users to have a deep understanding of its language and the ability to come up with tactics to construct effective proofs. This motivates the need for AI-assisted tactics generation, which can help users generate tactics for proving theorems in proof assistant more efficiently and effectively.

AI-assisted Proof Generation Recent advancements in AI, particularly NLP techniques, have shown great potential in enhancing formal theorem proving by automating proof generation and verification. AI-assisted proof generation leverages large language models (LLMs), which were trained on large databases of mathematical proofs, to assist LEAN users in constructing proofs by suggesting tactics, strategies, or even entire proofs based on the current proof state or goals.

AI-assisted proof generation opens up new possibilities for researchers and mathematicians, allowing them to tackle more complex problems with improved precision and confidence. This proposal aims to further explore and enhance the capabilities of AI-assisted proof generation for mathematical reasoning and verification, with a focus on combinatorial identities, linear algebra, and other mathematical domains (§ 4.2).

4 Proposed Research

The proposed research aims to develop Mathscape, an AI-enhanced and Mathscape integration theorem prover enhance mathematical reasoning and proof generation. We have developed a prototype of Mathscape and will use it as a foundation for the proposed research in expanding mathematical reasoning in new domains (§ 4.2), advancing AI/ML methodologies to aid proof generation (§ 4.3), and enhancing the usability of the LEAN proof assistant (§ 4.4).

4.1 Preliminary Work

Fig. 1 gives an overview of our Mathscape prototype, which consists of two main components: (1) an AI-assisted *synthesizer* that produces queries and tactics, and (2) a LEAN *processor* that interacts with the backend LEAN proof assistant. Mathscape is designed to provide AI-assisted proof generation capabilities to LEAN users, enabling them to construct proofs more effectively.

Example Here is a simple demonstration of Mathscape. The user puts in what they want to prove in natural language, e.g., *"prove that the sum of two even numbers is even"* to the AI synthesizer. The LLM-based synthesizer translates this into proper LEAN syntax, e.g., $\forall a, b \in \mathcal{N}.\texttt{even}(a) \cap \texttt{even}(b) \implies \texttt{even}(a+b)$ and sends this to the LEAN processor. The LEAN processor runs Lean,

which then asks the user to provide the proof for the statement. The LEAN process then sends this to AI synthesizer. The AI synthesizer parses and analyzes the feedback from the LEAN processor and generates a list of suggested tactics and proof steps, e.g., *"use the definition of even number and algebraic properties of additions"*, and translates this to LEAN syntax back to the LEAN processor, e.g., `def even`$(n : \mathcal{N}) : Prop := \exists k, n = 2 * k$ (a new definition) and `Nat.add_comm a b` (addition is commutative from Lean's library).

The LEAN processor then invokes LEAN to check the given tactics. LEAN can generate errors (e.g., syntax such as unknown keywords or variables or semantics such as type mismatching or incorrect/inappropriate use of tactics) or create subgoals to be proved, which are sent by the LEAN processor back to the AI synthesizer for further suggestions. This process iterates until the proof is completed (i.e., no goals remaining)or after some user-defined stopping criteria.

Note that these interactions (e.g., AI-generated proofs and feedback from Lean) are also recorded in natural language and can help the user understand the proof process and learn from it. This is particularly useful for non-experts who may not be familiar with Lean's syntax and tactics.

Prototype Our Mathscape prototype implements this iterative guess-and-check approach and uses OpenAI GPT-4 model for the AI synthesizer and LEAN 4. All interactions between the AI synthesizer and LEAN processor stored in a JSON database for further analysis and improvement. We have successfully applied the prototype to automatically prove several mathematical proofs and will build the proposed research on top of this prototype.

Note that the described interactive guess and check approach has been employed in existing AI-generated theorem proving techniques (e.g., Baldur [30], LeanDojo [106], AlphaGeometry [96] and many program analysis work (including those from PI Nguyen, e.g., [45,68,71]). The following proposed research will build on top this approach, identifies major challenges in each of the involved components of AI/FM/Maths, and develop techniques to tackle them.

4.2 Research Component 1: Extending Mathematical Reasoning

4.2.1 Domain 1: Combinatorics; a Cook's Tour

Combinatorics, the mathematics of counting or enumeration, is fundamental in all of mathematics (e.g., algebra, topology, partially ordered sets and geometry to name a few examples [4,32,90,91].) It has many applications in CS (e.g., complexity analysis, network design, cryptography) and other fields (e.g., resource allocations and scheduling, statistical samplings, DNA sequencing and strategy formulation in game theory [10, 12, 32, 38, 53, 67, 98]. Combinatorics often involves combinatorial identities, which represent the counting of elements in a set, e.g., the well-known *Pascal's Rule* $\binom{n}{k} = \binom{n-1}{k-1} + \binom{n-1}{k}$ shows the number of ways to choose k elements from n given elements in two different ways.

Combinatorial proof: In general, a *combinatorial identity* in our context means an equation $A(n_1, \ldots, n_k) = B(n_1, \ldots, n_k)$ that involves a given set of integer parameters $n_1, n_2, \ldots, n_k$ that holds true for any set of integers replacing the parameters n_i and (more importantly!) that has a combinatorial interpretation. Combinatorial identities can be proved using various proof techniques (e.g., algebra, induction, generating functions, and combinatorial proofs). Among these, *combinatorial proofs* are valued for their simplicity (least mathematical technicalities) and their ability to provide deeper insights into why certain mathematical statements are true. In this **RC**, we aim to

identify combinatorial identities that can be proved combinatorially. We also aim to formalize and prove them as well as existing combinatoric proofs using LEAN.

To do this, we can start with some well-known places where lists of known combinatorial identities can be found and where combinatorial proofs are elusive, for example a list written and posted by Richard Stanley [92, problems no. 17, 66, 85, 143, 238] to name a few. From such potential candidates, we will concentrate on binomial coefficients and also on the Stirling numbers of the second kind (see definition here below) because (a) the PI Agnarsson has done original work in this regards [1,2,8]. For example, a purely combinatorial proof of the identity $\sum_{i=1}^{n-k+1}\binom{n}{i}\left\{{n-i \atop k-1}\right\} = k\left\{{n \atop k}\right\}$ appears in [2]. Also, (b) the binomial coefficient have played, and still play, a fundamental role in combinatorial identities. Already in 1968 Donald Knuth asked the reader to: "Develop computer programs for simplifying sums that involve binomial coefficients" [43, Exercise 1.2.6.63]. This very exercise became the impetus for the writing of the entire expository book [83], a book describing how seemingly complicated identities involving binomial coefficients and other numbers can be verified, derived and proved often in a mechanical and routine way using techniques of hypergeometric series. These methods often yield identities that are seemingly impossible to explain in combinatorial way, even when the identities only involve positive integers and binomial coefficients. In other words, a combinatorial identity involving integers, binomial coefficients etc, can be valid even when it doesn't seem to have *any* combinatorial explanation. The book [83] is yet another place to look for combinatorial identities with no known combinatorial proofs.

Dissecting Identities A main part of this proposed work will be to dissect (or reverse engineer) the combinatorial identities into smaller parts, which can be proved combinatorially. This will help provide a deeper understanding of the underlying combinatorial structure and help us find combinatorial proofs for the identities. Such a dissection can however be challenging and many combinatorial identities remain unproven combinatorially (though they can be proved using different techniques, e.g., algebraic proofs, generating functions, hypergeometric series). Still, there are some approaches one can do address these challenges.

Firstly, when it is known that a certain identity has a proof that involves generating functions, then dissecting the very generating function that yields the desired identity, can sometimes be done to obtain a combinatorial proof. For example, a common proof using generating function techniques to prove the *Vandermonde's Identity* $\binom{m+n}{k} = \sum_{i=0}^{k}\binom{m}{i}\binom{n}{k-i}$ is obtained by looking at the coefficient of x^k in the expansion of $(x+1)^{m+n}$ using the binomial theorem on one hand, and then also in the expansion of the identical polynomial $(x+1)^m(x+1)^n$ on the other hand, where each factor is expanded using the binomial theorem.

Secondly, in some cases when coefficients or numbers that are defined in combinatorial ways appear in an identity, it can be fruitful to simply interpret both the left hand side and then the right hand side of this identity to obtain a set or a collection of which we count the number of certain elements. For example, consider the *Stirling numbers of the second kind* $\left\{{n \atop k}\right\}$ [4, Section 12.4], [32, Chapter 6], [90, p. 33][1]. They are usually defined combinatorially as "the number of ways to partition the set $\{1,\ldots,n\}$ into k non-empty unordered parts". They satisfy the following Pascal's-Rule-like identity: $\left\{{n \atop k}\right\} = k\left\{{n-1 \atop k}\right\} + \left\{{n-1 \atop k-1}\right\}$. A dissection of this identity for a combinatorial proof might proceed as follows:

(i) On the right we see partitions of two sets each containing $n-1$ elements. Assuming (for the moment) they are both $\{1,\ldots,n-1\}$, the former summand $k\left\{{n-1 \atop k}\right\}$ is the number of ways to

[1] the Stirling number of the fist kind are less used in combinatorics but more used in group theory

partition $\{1,\ldots,n-1\}$ into k parts and then designate (color it red) exactly one of these parts. The latter summand $\left\{{n-1 \atop k-1}\right\}$ is the number of ways to partition $\{1,\ldots,n-1\}$ into k parts.

(ii) Since a designation of (or assigning the color red to) a subset of $\{1,\ldots,n-1\}$ can be obtained by adding the element n to it, we see that the former summand counts the number of ways to partition $\{1,\ldots,n\}$ into k parts where n forms a part that has two or more elements and the latter summand counts the number of ways to partition $\{1,\ldots,n-1\}$ into k parts where n forms a part that has exactly one element, i.e. all the partitions of $\{1,\ldots,n\}$ into k nonempty parts.

A host of identities involving both binomial coefficients and Stirling numbers can be dissected in a similar fashion. There are other approaches for obtaining combinatorial proof (see 4.2.2.)

Hence, our plan is: (a) identify combinatorial identities that can be proven in a combinatorial way. Identifying them is realatively easy: any identity $A(n_1,\ldots,n_k) = B(n_1,\ldots,n_k)$ that involves combinatorially defined coefficients, like the binomial coefficients, Stirling numbers of the second kind etc can be viewed as a combinatorial identity. The real challenge is to find a combinatorial proof. But we did mention a few approaches that can be useful. (b) Use the existing definitions and theorems of finite sets, already implemented in LEAN to formalize and prove these combinatorial identities.

4.2.2 Domain 2: Bijective Proofs in Graph Theory

Simple graphs are one of the most important and practical tools for modeling practical and real-life problems (see [4, chapter 1] for more specifics.) By the discrete and finite nature of simple graphs, many proofs are especially well suited for computer verification, for example connectivity, acyclicity, cliques and both vertex and edge graph colorings.

Combinatorial or bijective proofs, in the spirit of previous component, are ubiquitous in graph theory as well. A prime example is a proof of *Cayley's Formula*: the number $\tau(K_n)$ of spanning trees of the complete labeled graph K_n is given by $\tau(K_n) = n^{n-2}$. Numerous elegant combinatorial proofs exist [9]. Despite their elegance, such proofs can be hard to establish, so there is big challenge in this regard. However, there are some approaches one can take here. The most comprehensive way to prove Cayley's Formula is by the *Matrix-Tree Theorem (MTT)*, which reduces computing $\tau(G)$ for any labeled simple graph G to that of computing a determinent [4, Sec. 4.5]. One special feature of the book [4] by PI Agnarsson (among many books on graph theory) is the unique and self-contained proof of the MTT using the *Cauchy–Binet formula* for determinants, which can in many instances be dissected to obtain alternative combinatorial derivations of $\tau(G)$, especially when G has a rich symmetric structure, for example when G is the complete labeled graph K_n or the complete labeled bipartite graph $K_{m,n}$.

Hence, our objective here is twofold: (a) to dissect known computational proofs (e.g. from the MTT) to obtain combinatorial proofs in graph theory. (b) By using the already present definitions and formulations for simple graphs already in LEAN we plan to add to library `mathlib` formal theorems in enumerative graph theory, which do not seem to have a strong presence[2] in LEAN.

4.3 Research Component 2: Enhancing AI/ML

The state-of-the-art LLM is not exclusively designed for the mathematical reasoning problems and hence sometimes cannot make good suggestions for solutions for these problems. In this section, we consider a class of mathematical problems which are inherently serial: it means that the problem

[2] as of May 2024

must be solved step-by-step, with each step dependent on the result of the previous steps. In this thrust, we aim to develop new AI/ML techniques to significantly improve the mathematical reasoning skills of LLMs by the following three approaches: (i) new prompting techniques, which are used to elicit the knowledge inside the pretrained LLMs such as GPT models; (ii) a new dataset with right supervision from LEAN, where LEAN provides the correct supervision of the LLMs and it is useful for more informative input for LLMs; (iii) a customized LLM for formal methods and mathematical reasoning, which is obtained via the new dataset with efficient fine-tuning techniques.

4.3.1 Customized Tree/Chain of Thoughts for Inherently Serial Problems

The goal of this task is to introduce a "Chain of Thought" [99] methodology to streamline and enhance the process of generating LEAN specifications for mathematical proofs. This methodology aims to improve the accuracy, efficiency, and comprehensibility of formal proofs by breaking down complex proofs into smaller, more manageable components. The concept of a customized tree or chain of thoughts for inherently serial problems represents a novel approach in computational problem-solving, particularly useful in contexts where decisions must be made sequentially and each choice impacts subsequent ones. This approach involves structuring the decision-making process as a tree or a linear chain, where each node or link represents a critical decision point or a thought process, tailored to the specific characteristics of the problem at hand. By mapping out these decisions in a hierarchical or sequential manner, one can systematically explore various strategies and outcomes, allowing for a more organized and efficient resolution of complex issues that require a step-by-step approach. This methodology is especially applicable in fields such as mathematical reasoning and algorithm design, where serial decision-making is crucial. The customization of the decision tree or chain to fit the unique demands of the problem enhances the ability to predict, analyze, and optimize outcomes, providing a clear pathway through intricate problem spaces. This chain of thought will be useful for helping LEAN to perform rigorous mathematical reasoning.

A Failure Example. The question is "Is any power of a chordal graph also chordal?" (here a graph is chordal if any cycle in the graph has a chord/edge, and by a k-th power of a graph G we mean the graph G^k on the same set of vertices as G but where any two vertices of distance k or less in G are connected with an edge in G^k. Here ChatGPT answered "YES" and then proceeded in giving a complicated but wrong proof. This is known not to be the case, as only odd powers of chordal graphs are chordal [5].

Motivated by the failure example, we propose a customized chain of thought technique for inherently serial problems which are difficult to solve. The key idea is to design the prompt optimization techniques to guide the LLMs to perform reasoning over *a path of dynamic programming*. There are two kinds of prompts which we will explore in the proposed work. (i) prompts that describe the size of the problem. In the context of the failure example, "the power of a chordal graph" is essentially the size of the problem. (ii) prompts that guide the LLM to think about a smaller scale of the problem. In the case of the failure example, we will design prompt to ask "Is power $n-1$ of a chordal graph is also chordal" before answering "Is power n of a chordal graph is also chordal?". The logic is similar to dynamic programming: when the LLM learns how to solve a problem with smaller scale correctly, it can help answer the problem with a slightly larger scale by one-step induction. We will leverage mathematical problems in graph theory (not covered by LEAN) to design such a chain of thought prompt automatically.

4.3.2 A New Dataset from Supervision from Lean

In this section, we aim to create a new dataset with the supervision from LEAN. The rationale is that LEAN can provide formal proofs for a proposition, which can serve as the groundtruth supervision signal and would have the potential for improve the LLMs due to the high quality curated dataset. However, the key challenge is that it requires a huge amount of annotation efforts for building such a high quality dataset. To address this challenge, we propose to leverage the active learning techniques to annotate the most informative questions with the formal LEAN proof, and use compositions of these true propositions (e.g., AND, OR, XOR operations) to generate new propositions and therefore increase the data size significantly.

The key idea of our proposed active data annotation is to design an algorithm to focus on *difficult but relevant* data samples. We propose to design a novel robust minimax formulation to address this issue. In particular, we propose a unified robust optimization problem which considers two tasks simultaneously: the querying of the annotations and the learning of the models. Therefore we will solve the following problem $\min_w \max_{p\in\Delta_n} \sum_{i=1}^n p_i \mathbb{E}_{Y|x_i}[\ell(w; x_i, Y)]$, where $\{x_1, \ldots, x_n\}$ are n unlabeled examples, w is the model parameter, and $\ell(w; x, y)$ is the loss function given the data (x, y), $E_{Y|x}$ denotes conditional expectation, $p = (p_1, \ldots, p_n)$ is the probability vector and $\Delta_n = \{p \in \mathbb{R}^n, p \geq 0, \sum_{i=1}^n p_i = 1, D(p, 1/n) \leq \rho\}$ is a constrained set which characterizes the uncertainty of the distribution, and $D(p, 1/n)$ denotes the probability distance between uniform distribution and distribution p. This optimization framework tries to learn the distribution p which corresponds to the most difficult training samples. In particular, we will actively label datasets from multiple domains, including Proof-Pile-2 [11], minif2f [110], and LeanDojo [106].

4.3.3 Customized LLMs for FM and Mathematical Reasoning

The need to customize a LLM for theorem proving arises from the unique and highly specialized nature of formal proofs, which requires domain-specific knowledge beyond the capabilities of general-purpose models. One of the biggest challenges we may face is the scarcity of data; the corpus of Lean-related content is relatively small compared to more generalized domains [11], at around 200 million tokens versus 6 trillions, limiting the model's ability to learn effectively from existing resources. To address this, other researchers have augmented the training corpus by including related scientific domains [11] and synthesizing formal mathematical data, such as generating new lemmas, theorems, and proofs [39, 84, 96]. Here, we aim to develop an advanced, customized LLM that leverages both domain-specific corpus and data synthesis techniques to enhance its ability in LEAN theorem proving.

Therefore, in this thrust, we will develop new multitask fine-tuning techniques for obtaining customized LLMs for mathematical reasoning with only small data. To tackle this issue, we suggest employing a meta-learning approach, as described by [29], for representation learning that facilitates effective adaptation using limited data. Representation learning, detailed by [16], enables automatic discovery of necessary features from raw data. In the context of mathematical theorem proving, we aim to use available mathematical data in related scientific domains and also generated new proofs to derive useful representations, enabling a linear predictor trained on this representation with downstream data to achieve high accuracy.

We consider m distributions $\{\mathcal{T}_i, i = 1, ..., m\}$ derived from a distribution $P_{\mathcal{T}}$. Each task's loss function is $\mathcal{L}(\mathbf{w}, \boldsymbol{\theta}_i, \xi)$, where $\mathbf{w}$ is the hyper-representation (meta learner) capturing data features across tasks, ξ represents the data, and $\boldsymbol{\theta}_i$ is each task's specific parameter for a base learner. The

goal is to find an optimal $\mathbf{w}$ that represents shared features effectively, minimizing the average loss across tasks and enabling quick adaptation of each base learner's parameter $\boldsymbol{\theta}_i$ to new tasks. This challenge is expressed as a bilevel optimization problem, where the base learner seeks to minimize its loss on the support set $\mathcal{S}_i$ using hyper-representation $\mathbf{w}$, while the meta learner optimizes $\mathbf{w}$ by evaluating all task-specific parameters $\boldsymbol{\theta}_i^*$ on their respective query sets $\mathcal{Q}_i$, where $i = 1, .., m$. Let $\boldsymbol{\theta} = (\boldsymbol{\theta}_1, ..., \boldsymbol{\theta}_m)$ be all the task-specific parameters, the objective function is the following:

$$\min_{\mathbf{w}} \frac{1}{m} \sum_{i=1}^{m} \frac{1}{|\mathcal{Q}_i|} \sum_{\xi \in \mathcal{Q}_i} \mathcal{L}(\mathbf{w}, \boldsymbol{\theta}_i^*(\mathbf{w}); \xi), \text{s.t.}, \boldsymbol{\theta}^*(\mathbf{w}) = \arg\min_{\boldsymbol{\theta}} \frac{1}{m} \sum_{i=1}^{m} \frac{1}{|\mathcal{S}_i|} \sum_{\zeta \in \mathcal{S}_i} \mathcal{L}(\mathbf{w}, \boldsymbol{\theta}_i; \zeta) + \frac{\mu}{2} \|\boldsymbol{\theta}_i\|^2, \quad (1)$$

where $\mathcal{S}_i$ and $\mathcal{Q}_i$ come from the task $\mathcal{T}_i$. In our proposed experiment, $\boldsymbol{\theta}_i$ is the parameter of the last linear layer of the neural network (e.g., the last layer of the pretrained foundation model such as Transformers), and $\mathbf{w}$ represents the other parameters except for the last layer. After a good representation $\hat{\mathbf{w}}$ is learned, we will freeze $\hat{\mathbf{w}}$ and update θ by the few samples of the downstream tasks. In the experiments, we will use public dataset to construct the tasks $\{\mathcal{T}_i, i = 1, ..., m\}$ and solve problem (1) to get $\hat{\mathbf{w}}_*$, and then learn $\hat{\theta} = \arg\min_\theta \frac{1}{|\mathcal{S}|} \sum_{\xi \in \mathcal{S}} \ell(\hat{\mathbf{w}}, \theta; \xi)$, where $\mathcal{S}$ is the set of training data from downstream task and $|\mathcal{S}|$ is small compared with number of samples in the source domain (i.e., $|\mathcal{T}_i|$, $i = 1, \ldots, m$). Therefore, the final foundation model we use for the prediction is $(\hat{\mathbf{w}}, \hat{\theta})$, where $\hat{\mathbf{w}}$ stands for the model parameter of the previous layers (meta learner parameter) and $\hat{\theta}$ stand for the last layer (base learner parameter). To solve the above problem, the off-the-shelf meta-learning algorithm [29] may not give satisfactory performance in the mathematical reasoning setting since it needs to capture long-sequence information in an autoregressive manner. To overcome the technical difficulty, we will design methods to update partial important parameters of $\mathbf{w}$ to capture the most relevant information. We will characterize which parameter to update by the coreset selection method and derive performance guarantees by leveraging analysis tools for meta learning, coreset selection and bilevel optimization recently by PI Liu's work [36, 37, 42].

4.4 Research Component 3 (Formal Methods): Improving Lean

Despite its powerful capabilities, LEAN could be further improved to be more accessible and useful. This RC focuses on improving LEAN in these aspects to better support the domains and applications in RC#1 and in general. Note that we treat LEAN as a black box and do not aim to improve the core LEAN system. Instead, we focus on the development of usability enhancements and interoperability with other systems. In addition, while we focus on LEAN, the ideas and methodologies developed in this RC can also be applied to other proof assistant systems, thereby contributing to the broader field of formal methods and mathematical reasoning.

4.4.1 LeanMatBridge: Integrating with Matlab

While LEAN is commonly used for mathematical reasoning, it lacks the ability to actually do "maths" as a dedicated software like MATLAB. To address this limitation, we will develop `LeanMatBridge` to integrate Mathscape/LEAN with MATLAB, a widely-used platform for numerical computing and data analysis. LeanMatBridge will enable Mathscape to leverage MATLAB's capabilities, while MATLAB users can benefit from Mathscape's formal verification capabilities.

LEAN leveraging Matlab Suppose the user wants to prove $x^2 - 2x + 1 \geq 0$, which might not be trivial because LEAN does not know how to factor a polynomial (even if it does, it might not do so efficiently). However, math systems such as MATLAB can easily compute the roots of the polynomial and obtain $(x-1)*(x-1)$. This result is then sent back to Lean, allowing it to prove the inequality through rewriting and simplification tactics.

LeanMatBridge will serves as an interface that communicates between LEAN and MATLAB. For example, it will convert Lean's proof goal into MATLAB, invoking MATLAB's functions, and then sending the results back to Lean. We will leverage LLM to translate the Lean's code into MATLAB and back.

One potential issue is the translation might be incorrect, e.g., we get $(x-1)*(x+1)$ instead of $(x-1)*(x-1)$. But here we can use LEAN itself to check the translation: i.e., proving that result $(x-1)*(x+1)$ is equal to $x^2 - 2x + 1$. While this requires additional effort, it ensures the correctness of the integration and checking the result of factoring is often easier than performing the factoring itself.

Here use factoring in as an example, but LeanMatBridge enables many scenarios that can leverage Matlab, e.g., combinatorics operations (factorial, permutations, binomial properties), graph algorithms (spanning trees, paths/cycles computations), and marix and linear equations operations.

Matlab Leveraging Lean Another application of this integration is that Matlab user can leverage LEAN to check their work. For example, they might develop some new factoring algorithm in Matlab and can use LEAN to "check" that the output is correct (e.g., the result obtained $(x-1)*(x-1)$ is indeed equal to x^2-2x+1). This can be useful in many scientific and engineering applications where formal verification and numerical computation are both needed.

4.4.2 Improving Lean's usability

FeedBackAnalyzer: Analyzing LEAN's output LEAN's feedback (e.g., error output) is crucial for constructing proofs. However, its feedback can sometimes be difficult to understand and misguide the AI synthesizer in Mathscape and confuse the user in understanding and correcting errors. For example, when using the popular *rewrite* tactic `rw [←sum_range_succ']` in LEAN to prove the lhs and rhs of

$$\sum_{i=1}^{k} \binom{m+1}{i}\binom{n}{k-i} + \binom{m+1}{0}\binom{n}{k} = \sum_{i=0}^{k} \binom{m+1}{i}\binom{n}{k-i}.$$

are the same, LEAN failed with the error: `tactic rewrite failed, did not find instance of the pattern in the target expression`. The user (e.g., a mathematician) can easily see that this pattern *is* already in the equation and get confused on why this occurs or how to fix it.

We will develop techniques and tools that analyze LEAN's feedback and provide more intuitive explanations and suggestions for improving the proof. We can start with a database of common patterns where LEAN fails to match and provide explanations and suggestions for fixing them. For example, the tool can explain why the pattern is not found in the target expression, or even better explain that it can only found a partial match but not entirely, e.g., cannot match the `sum` part on the lhs, the sum range is different ($i = 0$ vs $i = 1$), or if a pattern is almost matched but fails due to a small difference, the tool can highlight the discrepancy.

The tool will leverage AI and learns from pattern database and existing LEAN code to make suggestions, e.g., *"did you mean sum_range_succ" instead of sum_range_success"?*. The tool also

learns from the user's interaction with LEAN, e.g., what user did to fix error or respond to feedback from LEAN to provide better suggestions, e.g., expanding the range of the lhs so that $i = 0$. This proposed work will be part of our VSCode extension and leverages the existing LEAN LSP data (e.g., for code autocomplete) for better training and performance.

SmartSearch: Search Engine for LEAN Finding the right tactic or theorem is crucial to make progress in LEAN. However, the current structure and naming conventions of LEAN's tactics and theorems can be confusing and unintuitive. Moreover, the search capabilities of LEAN's API and `mathlib` contents are often unhelpful, returning many irrelevant results while missing the correct ones, e.g., searching for binomial, binomial theorem, or Pascal's rule returns many results *except* the correct ones (e.g., `add_pow` and `choose_suc_succ`) This makes it difficult for users to find the right tactics/theorems and for AI to learn from existing LEAN code.

To address this issue, we propose `LeanSmartSearch`, an advanced search tool designed to improve the discoverability of tactics and theorems within LEAN. `LeanSmartSearch` will leverage NLP and ML techniques to enhance the search experience in LEAN. `LeanSmartSearch` will index (or be trained on) Mathscape's documents and existing code (e.g., from theorems and tactics from `mathlib` and existing LEAN code and proofs). It will analyze the user's query to provide more accurate results, e.g., the document of `add_pow` mentioned binomial theorem and thus searching for "binomial theorem" will rank and return it, even if their names do not explicitly mention "binomial". In addition, `LeanSmartSearch` will match the query with the *contents* of the theorems and tactics, e.g., when the user search for Pascal rule, `LeanSmartSearch` will also consider its formula and description. This will help in identifying the correct resources even if the search terms do not exactly match the names or documents of the theorems or tactics.

Inspired by modern web search engines, `LeanSmartSearch` will include autocomplete and suggestions as users type their queries. This is done by `LeanSmartSearch` provides and ranks results based on the popular queries and commonly-used tactics and theorems. In addition, `LeanSmartSearch`'s search results will also include examples and related theorems and tactics (e.g., tactic `simp` is often used to handle complex equations by simplifying them). This will help users understand how to apply the resources they find and explore related concepts. Finally, to ensure seamless usability, `LeanSmartSearch` will be integrated with the LEAN VSCode extension, allowing users to search and use resultsf directly within their IDE.

4.5 Evaluation

To ensure the success and effectiveness of Mathscape, we will conduct a comprehensive evaluation encompassing several key aspects: comparison to existing tools, case studies, performance evaluation, scalability, and usability. Each component of the evaluation plan is designed to rigorously assess the advancements and contributions of our work.

Comparison to Existing Tools: We will benchmark Mathscape against existing proof assistants and AI-assisted theorem proving tools (e.g., LeanCoPilot and LeanDojo as mentioned in § 5). This includes comparing the performance of various components, e.g., LeanMatBridge, FeedBack-Analyzer, and SmartSearch, chain of thoughts search, with similar functionalities in other systems. Metrics such as proof completion rate, proof size (or number of iterations to complete proof), time taken to complete proofs, and accuracy of proof suggestions will be used to quantify the improvements. Additionally, we will evaluate the impact of our customized LLM on the overall efficiency and effectiveness of theorem proving in Lean.

Case Studies: We will conduct detailed case studies to demonstrate the practical applications and benefits of Mathscape in proposed domains, including combinatorial identities, graph theory, and formal verification in software engineering. Each case study will document the problem-solving process, the use of AI-assisted tools, and the outcomes achieved. By showcasing successful applications, we aim to illustrate the versatility and impact of our research.

Performance Evaluation: The performance of Mathscape will be evaluated through a series of controlled experiments. We will measure the computational efficiency, accuracy, and robustness of our AI-assisted proof generation techniques. Metrics such as proof generation time (e.g., number of iterations), error rates, and resource utilization will be collected and analyzed. Additionally, we will evaluate the effectiveness of our customized LLM in providing accurate and relevant proof suggestions, compared to baseline models.

Scalability: To assess the scalability of our system, we will evaluate its performance on a variety of proof tasks, ranging from simple to highly complex. This includes testing the system's ability to handle large datasets, extensive proof libraries, and high computational loads. We will also evaluate the scalability of our Lean3to4 and LeanMatBridge tools in terms of their ability to efficiently convert and integrate large volumes of code and data between different systems.

Usability: The usability of our enhanced LEAN system will be assessed through continuous user feedback and iterative improvements. We will integrate our tools with popular development environments, such as the LEAN VSCode extension, to provide a seamless user experience. Usability metrics will include the intuitiveness of the interface, the clarity of error messages and feedback, and the overall workflow efficiency. We will also gather feedback on the utility and effectiveness of our educational materials, such as the online course module and interactive tutorials.

User Studies: To assess the usability and accessibility of our enhanced LEAN system, we will conduct user studies involving participants from diverse backgrounds, including students, researchers, and industry professionals. These studies will focus on the ease of use, learning curve, and overall user experience. Participants will be asked to complete a series of proof tasks using both the original and enhanced versions of Lean, and their feedback will be collected through surveys and interviews. Key metrics will include user satisfaction, perceived ease of use, and the time required to learn and apply the new tools.

We note that while we do not have much experience in conducting user studies, we believe they are essential for evaluating the usability and effectiveness of Mathscape, especially since our target users include non-experts, undergrad students, and researchers from diverse backgrounds. As such, we will seek collaboration and guidance from experts in HCI and user experience research to design and execute these studies effectively.

5 Related works

LEAN improvements There has been several ongoing projects focus on improving the usability of LEAN. Loogle[3] is a search engine that allows users to search for theorems implemented in LEAN based on various syntax expressions. LeanCoPilot [89] is a VSCode extension that uses LLMs to automatically suggest new proof steps for users. To provide an interface for interactions between LEAN compiler and automatic proof generation systems, several works have also been introduced, including LeanStep [35], `lean-gym` [84] and LeanDojo [106]. These interfaces work by wrapping the LEAN compiler and proof generation systems in a REPL (read-eval-print loop) environment.

[3] `https://github.com/nomeata/loogle`

AI-assisted Formal Theorem Proving Many recent advancements to the general field of AI-assisted formal theorem proving can be categorized into autoformalization and proof generation. Autoformalization focuses on converting manually written mathematical theorems and proofs into a verifiable program automatically [51]. This can be done by formulating this task as a variant of machine translation problem [97] or leveraging the recent developments of LLMs [11, 88, 102]. Proof generation is the core problem for theorem proving, which aims to generate the proof of the theorem in a formal language. Various approaches have been proposed for this topic, such as incorporating syntax of formal languages [105], combining with conditional language modeling of LLMs [30, 84, 85], or jointly training with other tasks including autoformalization [103] and premise selection [106]. One notable research here is AlphaGeometry [96] where the authors synthesize a dataset of International Math Olympiad geometry problems with proofs.

6 Project Plan

Our estimate time for the development of Mathscape is 36 months. PI Agnarsson will lead the development of RC1 on formalizing and extending to combinatorics and graph theory, PI Liu will lead the development of RC2 on enhancing ML/LLM's capabilities for these domains, and PI Nguyen will lead the development of RC3 on improving LEAN's usability. Each RC will have a dedicated GRA and undergrad students to support the development and evaluation of Mathscape.

We note that while each PI will focus on their respective RC (and we estimate about 8 – 10 months on developments per RC and 2 months for evaluation and analyzing results), they will collaborate closely to ensure the success of the project. We also will co-advise the GRAs and students to ensure that they have a broad understanding of the project and can contribute to all aspects of the project.

As mentioned in § 7, we will integrate Mathscape in our teaching and thus Mathscape will be continuously evaluated and improved based on feedback of students and software developers, who often take our courses. The project will support three GRAs (mentoring plan attached) and multiple undergrad students. We will also seek REU supplements as needed to support more students.

7 Education Integration

7.1 Curriculum Development and Mathscape Starter Book

We will integrate this research into our relevant courses in Math, AI/ML, and Software Analysis. For example, in PI Nguyen's Software Construction and Specification course, in addition to using Z3 SMT solving for Hoare verification, students will be introduced to AI-enabled Formal Method tools, e.g., Lean-based Mathscape, to encode and prove software specification and conjectures/theorems in algebra. GMU is creating a new MS program focusing on AI/ML Reasoning and Safety program, and the PIs will develop a course on AI safety in which Mathscape will be adapted to demonstrate interactive reasoning of desired properties of DNNs. Over the time frame of this proposal, we will teach these courses multiple times (avg. 40 students/class) and thus have ample opportunities to evaluate and improve materials.

We will also collaborate to develop a math book that leverages AI to aid mathematical reasoning and, naturally enough, uses Mathscape as a case study. Unlike a traditional math book, this will be written in Markdown/Python notebook format and contains "live" code that can be modified and

executed by the students (e.g., similar to the Fuzzing book [107]). It will also have tutorials, code examples, and exercises that guide students in using Mathscape to proving mathematical theorems and conjectures. PI Agnarsson, who has written a widely-used textbook, *"Graph theory: modeling, applications, and algorithms" [4]*, will lead the development of the book.

The book will be used in our courses. PI Nguyen has experience in working with Wiley to publish his course and will explore the possibility of publishing the book and an associated course with a major publisher to reach a wider audience. Many of our collaborators who teach math/formal method/ML courses are potential users and contributors to this open-source book.

7.2 Undergraduate Research and Outreach

Fig. 2: Liu (left) and his PhD and three ASSIP HS students.

The PIs have been actively involved in mentoring undergraduate and high school students in research and are excited to continue doing so in this project. PI Nguyen is working with two female undergraduates and in the past has worked with more than 15 undergraduate students and published multiple papers with them [40, 41, 68, 69, 75–78]. PI Liu has advised multiple high school students from the GMU Aspiring Scientists Summer Internship Program (ASSIP) from Summer 2022 (Fig. 2) to Summer 2023, including one female high school student (Marina Lin from Thomas Jefferson High School for Science and Technology). PI Agnarsson has directed Senior Honors Theses and served as the Project Chair for the *Mason Experimental Geometry Lab (MEGL)* undergrad projects at GMU. He also frequently gives invited talks at various high schools in the northern Virginia area. (e.g. an invited talk for the Math Honor Society at Westfield High School.)

This project already has an undergrad student working on it, and we will continue to recruit and mentor undergraduate and highshool students to work on it. Success will be measured by the number of students involved, the quality of their contributions (e.g., commits to to Github repo hosting the project, paper submissions), and their career paths.

8 Results from Prior NSF Support

PI Nguyen's most related NSF support is #2238133 *CAREER: NeuralSAT: A Constraint-Solving Framework for Verifying Deep Neural Networks*, $510,509.00, 7/1/2023–6/30/2028. Intellectual Merit: this project is on the development, testing, and optimization of the NeuralSAT DNN tool. This project also aims to test Mathscape using metamorphic testing used to stress-test NeuralSAT. Broader Impacts: This work aims to produce scalable and reliable theories and methods for analyzing DNNs and therefore ensuring AI safety. PI Nguyen ks working with a graduate student on this project and has produced one publication [28].

PIs Agnarsson and Liu have no prior NSF support.

References

[1] G. Agnarsson. Induced subgraphs of hypercubes. *European J. Combin.*, 34(2):155–168, 2013.

[2] G. Agnarsson. On a special class of hyper-permutahedra. *Electron. J. Combin.*, 24(3):Paper No. 3.46, 25, 2017.

[3] G. Agnarsson and N. Epstein. On monomial ideals and their socles. *Order*, 37(2):341–369, 2020.

[4] G. Agnarsson and R. Greenlaw. *Graph theory: modeling, applications, and algorithms*. Pearson Prentice Hall, Upper Saddle River, NJ, 2007.

[5] G. Agnarsson, R. Greenlaw, and M. M. Halldórsson. On powers of chordal graphs and their colorings. *Congr. Numer.*, 144:41–65, 2000.

[6] G. Agnarsson, M. M. Halldórsson, and E. Losievskaja. SDP-based algorithms for maximum independent set problems on hypergraphs. *Theoret. Comput. Sci.*, 470:1–9, 2013.

[7] G. Agnarsson and J. Lawrence. Power-closed ideals of polynomial and laurent polynomial rings. *J. Pure Appl. Algebra*, page to appear, 2024.

[8] G. Agnarsson and W. D. Morris. On Minkowski sums of simplices. *Ann. Comb.*, 13(3):271–287, 2009.

[9] M. Aigner and G. M. Ziegler. *Proofs from The Book*. Springer, Berlin, sixth edition, 2018. See corrected reprint of the 1998 original [MR1723092], Including illustrations by Karl H. Hofmann.

[10] S. Arora and B. Barak. *Computational complexity*. Cambridge University Press, Cambridge, 2009. A modern approach.

[11] Z. Azerbayev, H. Schoelkopf, K. Paster, M. D. Santos, S. McAleer, A. Q. Jiang, J. Deng, S. Biderman, and S. Welleck. Llemma: An open language model for mathematics. *arXiv preprint arXiv:2310.10631*, 2023.

[12] R. Balakrishnan and S. Sridharan. *Discrete mathematics—graph algorithms, algebraic structures, coding theory, and cryptography*. CRC Press, Boca Raton, FL, 2020.

[13] Y. Bao, M. Crawshaw, S. Luo, and M. Liu. Fast composite optimization and statistical recovery in federated learning. In *International Conference on Machine Learning*, pages 1508–1536. PMLR, 2022.

[14] Y. Bao, A. Shehu, and M. Liu. Global convergence analysis of local sgd for two-layer neural network without overparameterization. *Advances in Neural Information Processing Systems*, 36, 2023.

[15] B. Barras, S. Boutin, C. Cornes, J. Courant, Y. Coscoy, D. Delahaye, D. de Rauglaudre, J.-C. Filliâtre, E. Giménez, H. Herbelin, et al. The coq proof assistant reference manual. *INRIA, version*, 6(11), 1999.

[16] Y. Bengio, A. Courville, and P. Vincent. Representation learning: A review and new perspectives. *IEEE transactions on pattern analysis and machine intelligence*, 35(8):1798–1828, 2013.

[17] D. Beyer. Progress on software verification: Sv-comp 2022. In *International Conference on Tools and Algorithms for the Construction and Analysis of Systems*, pages 375–402. Springer, 2022.

[18] C. Brix, S. Bak, C. Liu, and T. T. Johnson. The fourth international verification of neural networks competition (vnn-comp 2023): Summary and results, 2023.

[19] D. Castelvecchi. Mathematicians welcome computer-assisted proof in 'grand unification'theory. *Nature*, 595(7865):18–19, 2021.

[20] D. Castelvecchi. Are chatgpt and alphacode going to replace programmers? *Nature*, 2022.

[21] S. Chakraborty, S. K. Lahiri, S. Fakhoury, M. Musuvathi, A. Lal, A. Rastogi, A. Senthilnathan, R. Sharma, and N. Swamy. Ranking llm-generated loop invariants for program verification. *arXiv preprint arXiv:2310.09342*, 2023.

[22] M. Crawshaw, Y. Bao, and M. Liu. EPISODE: Episodic gradient clipping with periodic resampled corrections for federated learning with heterogeneous data. In *The Eleventh International Conference on Learning Representations*, 2023.

[23] M. Crawshaw, Y. Bao, and M. Liu. Federated learning with client subsampling, data heterogeneity, and unbounded smoothness: A new algorithm and lower bounds. *Advances in Neural Information Processing Systems*, 36, 2023.

[24] M. Crawshaw, M. Liu, F. Orabona, W. Zhang, and Z. Zhuang. Robustness to unbounded smoothness of generalized signsgd. *Advances in Neural Information Processing Systems*, 2022.

[25] X. Cui, W. Zhang, A. Kayi, M. Liu, U. Finkler, B. Kingsbury, G. Saon, and D. Kung. Asynchronous decentralized distributed training of acoustic models. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 29:3565–3576, 2021.

[26] L. De Moura and N. Bjørner. Z3: An efficient SMT Solver. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 4963 LNCS:337–340, 2008.

[27] L. De Moura, S. Kong, J. Avigad, F. Van Doorn, and J. von Raumer. The lean theorem prover (system description). In *Automated Deduction-CADE-25: 25th International Conference on Automated Deduction, Berlin, Germany, August 1-7, 2015, Proceedings 25*, pages 378–388. Springer, 2015.

[28] H. Duong, D. Xu, T. Nguyen, and M. Dwyer. Harnessing Neuron Stability to Improve DNN Verification. *Proceedings of the ACM on Software Engineering (PACMSE)*, (FSE):to appear, 2024.

[29] C. Finn, P. Abbeel, and S. Levine. Model-agnostic meta-learning for fast adaptation of deep networks. In *International conference on machine learning*, pages 1126–1135. PMLR, 2017.

[30] E. First, M. Rabe, T. Ringer, and Y. Brun. Baldur: Whole-proof generation and repair with large language models. In *Proceedings of the 31st ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering*, pages 1229–1241, 2023.

[31] S. Forrest, T. Nguyen, W. Weimer, and C. Le Goues. A genetic programming approach to automated software repair. In *Proceedings of the 11th Annual conference on Genetic and evolutionary computation*, pages 947–954, 2009.

[32] R. L. Graham, D. E. Knuth, and O. Patashnik. *Concrete mathematics*. Addison-Wesley Publishing Company, Reading, MA, second edition, 1994. A foundation for computer science.

[33] Y. Guo, M. Liu, T. Yang, and T. Rosing. Improved schemes for episodic memory-based lifelong learning. *Advances in Neural Information Processing Systems*, 33, 2020.

[34] Z. Guo, M. Liu, Z. Yuan, L. Shen, W. Liu, and T. Yang. Communication-efficient distributed stochastic auc maximization with deep neural networks. *International Conference on Machine Learning*, 2020.

[35] J. M. Han, J. Rute, Y. Wu, E. Ayers, and S. Polu. Proof artifact co-training for theorem proving with language models. In *International Conference on Learning Representations*, 2022.

[36] J. Hao, X. Gong, and M. Liu. Bilevel optimization under unbounded smoothness: A new algorithm and convergence analysis. In *The Twelfth International Conference on Learning Representations*, 2024.

[37] J. Hao, K. Ji, and M. Liu. Bilevel coreset selection in continual learning: A new formulation and algorithm. *Advances in Neural Information Processing Systems*, 36, 2023.

[38] T. Harks and J. Schwarz. A unified framework for pricing in nonconvex resource allocation games. *SIAM J. Optim.*, 33(2):1223–1249, 2023.

[39] Y. Huang, X. Lin, Z. Liu, Q. Cao, H. Xin, H. Wang, Z. Li, L. Song, and X. Liang. Mustard: Mastering uniform synthesis of theorem and proof data. *arXiv preprint arXiv:2402.08957*, 2024.

[40] D. Ishimwe, K. Nguyen, and T. Nguyen. Dynaplex: analyzing program complexity using dynamically inferred recurrence relations. *Proc. ACM Program. Lang.*, 5(OOPSLA):1–23, 2021.

[41] D. Ishimwe, T. Nguyen, and K. Nguyen. Dynaplex: Inferring asymptotic runtime complexity of recursive programs. In *2022 IEEE/ACM 44th International Conference on Software Engineering: Companion Proceedings (ICSE-Companion)*, pages 61–64. IEEE, 2022.

[42] K. Ji, M. Liu, Y. Liang, and L. Ying. Will bilevel optimizers benefit from loops. *Advances in Neural Information Processing Systems*, 2022.

[43] D. E. Knuth. *The art of computer programming. Vol. 1: Fundamental algorithms*. Addison-Wesley Publishing Co., Reading, Mass.-London-Don Mills, Ont., 1969. Second printing.

[44] P. Lammich. Generating verified llvm from isabelle/hol. In *10th International Conference on Interactive Theorem Proving (ITP 2019)*. Schloss Dagstuhl-Leibniz-Zentrum für Informatik, 2019.

[45] T. C. Le, T. Antonopoulos, P. Fathololumi, E. Koskinen, and T. Nguyen. Dynamite: dynamic termination and non-termination proofs. *Proceedings of the ACM on Programming Languages*, 4(OOPSLA):1–30, 2020.

[46] C. Le Goues, T. Nguyen, S. Forrest, and W. Weimer. Genprog: A generic method for automatic software repair. *Ieee transactions on software engineering*, 38(1):54–72, 2011.

[47] Lean Community. 100 theorems, 2024.

[48] Y. Lei, M. Liu, and Y. Ying. Generalization guarantee of SGD for pairwise learning. In *Advances in Neural Information Processing Systems*, 2021.

[49] X. Li, M. Liu, and F. Orabona. On the initialization for convex-concave min-max problems. In *Algorithmic Learning Theory*, 2022.

[50] Y. Li, D. Choi, J. Chung, N. Kushman, J. Schrittwieser, R. Leblond, T. Eccles, J. Keeling, F. Gimeno, A. Dal Lago, et al. Competition-level code generation with alphacode. *Science*, 378(6624):1092–1097, 2022.

[51] Z. Li, J. Sun, L. Murphy, Q. Su, Z. Li, X. Zhang, K. Yang, and X. Si. A survey on deep learning for theorem proving. *arXiv preprint arXiv:2404.09939*, 2024.

[52] Q. Lin, M. Liu, H. Rafique, and T. Yang. Solving weakly-convex-weakly-concave saddle-point problems as weakly-monotone variational inequality. *arXiv preprint arXiv:1810.10207*, 2018.

[53] C. L. Liu. A combinatorial study of some scheduling algorithms. In *Applied computation theory: analysis, design, modeling*, Prentice-Hall Series in Automatic Computation, pages 82–105. Prentice-Hall, Inc., Englewood Cliffs, NJ, 1976.

[54] M. Liu, Z. Li, X. Wang, J. Yi, and T. Yang. Adaptive negative curvature descent with applications in non-convex optimization. In *Advances in Neural Information Processing Systems*, pages 4853–4862, 2018.

[55] M. Liu, Y. Mroueh, J. Ross, W. Zhang, X. Cui, P. Das, and T. Yang. Towards better understanding of adaptive gradient algorithms in generative adversarial nets. In *International Conference on Learning Representations*, 2020.

[56] M. Liu and F. Orabona. On the initialization for convex-concave min-max problems. In *Algorithmic Learning Theory*, 2022.

[57] M. Liu and T. Yang. Adaptive accelerated gradient converging method under h\"{o} lderian error bound condition. In *Advances in Neural Information Processing Systems*, pages 3104–3114, 2017.

[58] M. Liu, Z. Yuan, Y. Ying, and T. Yang. Stochastic auc maximization with deep neural networks. *ICLR*, 2020.

[59] M. Liu, W. Zhang, Y. Mroueh, X. Cui, T. Yang, and P. Das. A decentralized parallel algorithm for training generative adversarial nets. *Advances in Neural Information Processing Systems*, 2020.

[60] M. Liu, X. Zhang, Z. Chen, X. Wang, and T. Yang. Fast stochastic auc maximization with $o(1/n)$-convergence rate. In *International Conference on Machine Learning*, pages 3195–3203, 2018.

[61] M. Liu, X. Zhang, L. Zhang, R. Jin, and T. Yang. Fast rates of erm and stochastic approximation: Adaptive to error bound conditions. In *Advances in Neural Information Processing Systems*, pages 4678–4689, 2018.

[62] M. Liu, X. Zhang, X. Zhou, and T. Yang. Faster online learning of optimal threshold for consistent f-measure optimization. In *Advances in Neural Information Processing Systems*, pages 3889–3899, 2018.

[63] M. Liu, Z. Zhuan, Y. Lei, and C. Liao. A communication-efficient distributed gradient clipping algorithm for training deep neural networks. In *Advances in Neural Information Processing Systems*, 2022.

[64] B. Mariano, J. Reese, S. Xu, T. Nguyen, X. Qiu, J. S. Foster, and A. Solar-Lezama. Program synthesis with algebraic library specifications. *Proceedings of the ACM on Programming Languages*, 3(OOPSLA):1–25, 2019.

[65] L. d. Moura and N. Bjørner. Z3: An efficient smt solver. In *International conference on Tools and Algorithms for the Construction and Analysis of Systems*, pages 337–340. Springer, 2008.

[66] L. d. Moura and S. Ullrich. 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 28*, pages 625–635. Springer, 2021.

[67] M. Newman. *Networks*. Oxford University Press, Oxford, second edition, 2018.

[68] K. Nguyen and T. Nguyen. Gentree: Using decision trees to learn interactions for configurable software. In *2021 IEEE/ACM 43rd International Conference on Software Engineering (ICSE)*, pages 1598–1609. IEEE, 2021.

[69] K. Nguyen, T. Nguyen, and Q.-S. Phan. Analyzing the cmake build system. In *2022 IEEE/ACM 44th International Conference on Software Engineering: Software Engineering in Practice (ICSE-SEIP)*, pages 27–28. IEEE, 2022.

[70] T. Nguyen. The DIG Invariant Generation System. `https://github.com/dynaroars/dig/`, last accessed May 28, 2024.

[71] T. Nguyen, T. Antonopoulos, A. Ruef, and M. Hicks. Counterexample-guided approach to finding numerical invariants. In *Foundations of Software Engineering*, pages 605–615, 2017.

[72] T. Nguyen, M. B. Dwyer, and W. Visser. Symlnfer: Inferring program invariants using symbolic states. In *2017 32nd IEEE/ACM International Conference on Automated Software Engineering (ASE)*, pages 804–814. IEEE, 2017.

[73] T. Nguyen, D. Kapur, W. Weimer, and S. Forrest. Using dynamic analysis to discover polynomial and array invariants. In *International Conference on Software Engineering (ICSE)*, pages 683–693. IEEE, 2012.

[74] T. Nguyen, D. Kapur, W. Weimer, and S. Forrest. Dig: a dynamic invariant generator for polynomial and array invariants. *ACM Transactions on Software Engineering and Methodology (TOSEM)*, 23(4):1–30, 2014.

[75] T. Nguyen and K. Nguyen. Using symbolic execution to analyze linux kbuild makefiles. In *2020 IEEE International Conference on Software Maintenance and Evolution (ICSME)*, pages 712–716. IEEE, 2020.

[76] T. Nguyen and K. Nguyen. Using symbolic execution to analyze linux kbuild makefiles. In *International Conference on Software Maintenance and Evolution*, pages 712–716. IEEE, 2020.

[77] T. Nguyen, K. Nguyen, and H. Duong. Syminfer: Inferring numerical invariants using symbolic states. In *2022 IEEE/ACM 44th International Conference on Software Engineering: Companion Proceedings (ICSE-Companion)*, pages 197–201. IEEE, 2022.

[78] T. Nguyen, K. H. Nguyen, and M. Dwyer. Using symbolic states to infer numerical invariants. *IEEE Transactions on Software Engineering*, 2021.

[79] T. Nguyen, W. Weimer, D. Kapur, and S. Forrest. Connecting program synthesis and reachability: Automatic program repair using test-input generation. In *International Conference on Tools and Algorithms for the Construction and Analysis of Systems*, pages 301–318. Springer, 2017.

[80] T.-D. Nguyen, T. Le-Cong, T. H. Nguyen, X.-B. D. Le, and Q.-T. Huynh. Toward the analysis of graph neural networks. In *2022 IEEE/ACM 44rd International Conference on Software Engineering-New Ideas and Emerging Results (ICSE-NIER)*, 2022.

[81] L. C. Paulson. *Isabelle: A generic theorem prover*. Springer, 1994.

[82] K. Pei, D. Bieber, K. Shi, C. Sutton, and P. Yin. Can large language models reason about program invariants? In *International Conference on Machine Learning*, pages 27496–27520. PMLR, 2023.

[83] M. Petkovšek, H. S. Wilf, and D. Zeilberger. $A = B$. A K Peters, Ltd., Wellesley, MA, 1996. With a foreword by Donald E. Knuth, With a separately available computer disk.

[84] S. Polu, J. M. Han, K. Zheng, M. Baksys, I. Babuschkin, and I. Sutskever. Formal mathematics statement curriculum learning. *arXiv preprint arXiv:2202.01344*, 2022.

[85] S. Polu and I. Sutskever. Generative language modeling for automated theorem proving. *arXiv preprint arXiv:2009.03393*, 2020.

[86] Project NeuralSAT. The NeuralSAT solver for Verifying DNNs, 2022. `https://github.com/dynaroars/neuralsat`.

[87] H. Rafique, M. Liu, Q. Lin, and T. Yang. Non-convex min-max optimization: Provable algorithms and applications in machine learning. *arXiv preprint arXiv:1810.02060*, 2018.

[88] Z. Shao, P. Wang, Q. Zhu, R. Xu, J. Song, M. Zhang, Y. Li, Y. Wu, and D. Guo. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300*, 2024.

[89] P. Song, K. Yang, and A. Anandkumar. Towards large language models as copilots for theorem proving in Lean. *arXiv preprint arXiv: Arxiv-2404.12534*, 2024.

[90] R. P. Stanley. *Enumerative combinatorics. Vol. 1*, volume 49 of *Cambridge Studies in Advanced Mathematics*. Cambridge University Press, Cambridge, 1997. With a foreword by Gian-Carlo Rota, Corrected reprint of the 1986 original.

[91] R. P. Stanley. *Enumerative combinatorics. Vol. 2*, volume 62 of *Cambridge Studies in Advanced Mathematics*. Cambridge University Press, Cambridge, 1999. With a foreword by Gian-Carlo Rota and appendix 1 by Sergey Fomin.

[92] R. P. Stanley. Bijective proof problems, 2009. `https://math.mit.edu/~rstan/bij.pdf`, last accessed May 28, 2024.

[93] Tereance Tao. A digitisation of the proof of the Polynomial Freiman-Ruzsa Conjecture in Lean 4, 2024.

[94] ThanhVu Nguyen and Leonardo de Moura. Z3 Python Interface, 2022. `https://github.com/Z3Prover/z3/blob/master/src/api/python/z3/z3util.py`, last accessed May 28, 2024.

[95] ThanhVu Nguyen and Timos Antonopoulos. NLA-DIGBENCH, 2022. `https://gitlab.com/sosy-lab/benchmarking/sv-benchmarks/-/tree/main/c/nla-digbench`, last accessed May 28, 2024.

[96] T. H. Trinh, Y. Wu, Q. V. Le, H. He, and T. Luong. Solving olympiad geometry without human demonstrations. *Nature*, 625(7995):476–482, 2024.

[97] Q. Wang, C. Brown, C. Kaliszyk, and J. Urban. Exploration of neural machine translation in autoformalization of mathematics in mizar. In *Proceedings of the 9th ACM SIGPLAN International Conference on Certified Programs and Proofs*, pages 85–98, 2020.

[98] S. Wang, Y. Gao, Y. Zhou, B. Pan, X. Xu, and T. Li. Reducing the statistical error of generative adversarial networks using space-filling sampling. *Stat*, 13(1):Paper No. e655, 15, 2024.

[99] J. Wei, X. Wang, D. Schuurmans, M. Bosma, F. Xia, E. Chi, Q. V. Le, D. Zhou, et al. Chain-of-thought prompting elicits reasoning in large language models. *Advances in neural information processing systems*, 35:24824–24837, 2022.

[100] W. Weimer, T. Nguyen, C. Le Goues, and S. Forrest. Automatically finding patches using genetic programming. In *2009 IEEE 31st International Conference on Software Engineering*, pages 364–374. IEEE, 2009.

[101] S. Welleck and R. Saha. Llmstep: Llm proofstep suggestions in lean. *arXiv preprint arXiv:2310.18457*, 2023.

[102] Y. Wu, A. Q. Jiang, W. Li, M. Rabe, C. Staats, M. Jamnik, and C. Szegedy. Autoformalization with large language models. *Advances in Neural Information Processing Systems*, 35:32353–32368, 2022.

[103] H. Xin, H. Wang, C. Zheng, L. Li, Z. Liu, Q. Cao, Y. Huang, J. Xiong, H. Shi, E. Xie, et al. Lego-prover: Neural theorem proving with growing libraries. *arXiv preprint arXiv:2310.00656*, 2023.

[104] Y. Xu, M. Liu, Q. Lin, and T. Yang. Admm without a fixed penalty parameter: faster convergence with new adaptive penalization. In *Advances in Neural Information Processing Systems*, pages 1267–1277, 2017.

[105] K. Yang and J. Deng. Learning to prove theorems via interacting with proof assistants. In *International Conference on Machine Learning*, pages 6984–6994. PMLR, 2019.

[106] K. Yang, A. Swope, A. Gu, R. Chalamala, P. Song, S. Yu, S. Godil, R. J. Prenger, and A. Anandkumar. Leandojo: Theorem proving with retrieval-augmented language models. *Advances in Neural Information Processing Systems*, 36, 2024.

[107] A. Zeller, R. Gopinath, M. Böhme, G. Fraser, and C. Holler. The fuzzing book, 2019. `https: https://www.fuzzingbook.org`, last accessed May 28, 2024.

[108] G. Zheng, T. Nguyen, S. G. Brida, G. Regis, M. F. Frias, N. Aguirre, and H. Bagheri. Flack: Counterexample-guided fault localization for alloy models. In *2021 IEEE/ACM 43rd International Conference on Software Engineering (ICSE)*, pages 637–648. IEEE, 2021.

[109] G. Zheng, T. Nguyen, S. Gutiérrez Brida, G. Regis, M. Frias, N. Aguirre, and H. Bagheri. ATR: Template-based Repair for Alloy Specifications. In *Proceedings of the 31st ACM SIGSOFT International Symposium on Software Testing and Analysis*, pages 666–677, 2022.

[110] K. Zheng, J. M. Han, and S. Polu. Minif2f: a cross-system benchmark for formal olympiad-level mathematics. *arXiv preprint arXiv:2109.00110*, 2021.

[111] X. Zhu, J. Hao, Y. Guo, and M. Liu. Auc maximization in imbalanced lifelong learning. In *Uncertainty in Artificial Intelligence*, pages 2574–2585. PMLR, 2023.