A Lean Dataset for International Math Olympiad: Small Steps towards Writing Math Proofs for Hard Problems

Anonymous authors
Paper under double-blind review

Abstract

Using AI to write formal proofs for mathematical problems is a challenging task that has seen some advancements in recent years. Automated systems such as Lean can verify the correctness of proofs written in formal language, yet writing the proofs in formal language can be challenging for humans and machines. The miniF2F benchmark has 20 IMO problems in its testing set, yet formal proofs are available only for 6 of these problems (3 of which are only written by mathematicians). The model with best accuracy can only prove 4 of these 20 IMO problems, from 1950s and 60s, while its training set is a secret. In this work, we write complete, original formal proofs for the remaining 13 IMO problems in Lean along with 3 extra problems from IMO 2022 and 2023. This effort expands the availability of proof currently in the public domain by creating 5,150 lines of Lean proof. The goal of the paper is to pave the way for developing AI models that can automatically write the formal proofs for all the IMO problems in miniF2F and beyond. In this pursuit, we devise a method to decompose the proof of these problems into their building blocks, constructing a dataset of about 900 lemmas with 25,500 lines of Lean code. These lemmas are not trivial, yet they are approachable, providing the opportunity to evaluate and diagnose the failures and successes of AI models. We then evaluate the ability of o1-mini in writing formal proofs for these lemmas with zero shot prompting, CoT reasoning and lemma retrieval. Our dataset serves as a stepping stone towards proving IMO problems.

1 Introduction

Machine learning methods have made significant progress in many different domains. This progress can often be described as the automation of learning from data. Availability of high quality data is crucial for the community to develop better models and methods. ImageNet (Deng et al., 2009), for example, played a significant role in the development of ML methods in computer vision. More recent models such as GPT and Llama rely on training set purportedly almost as large as the entire contents of the Web. On the other hand, the failures of these models on topics such as math are often explained via lack of high-quality materials on those topics in their training set. This issue is partially addressed by curation of relevant data, e.g., Lightman et al. (2023) recruits human labelers and AlphaGeometry (Trinh et al., 2024) generates synthetic data. For the topic of automated theorem proving in formal language, high quality data is so scarce, to the extent that the formal proofs are not available for almost half of the problems in the miniF2F dataset, the benchmark of the community. So, when automated methods fail to write a correct proof for the problems in the benchmark dataset, it is not clear where they have failed, and what the correct answer looks like. In this work, we study 16 IMO problems, in number theory and algebra. 13 of the 16 problems are from the miniF2F dataset and 12 of the 16 do not have a formal proof available in the public domain. Moreover, for all the 40 IMO problems in the miniF2F dataset, we contribute the complete, original, formal proofs in Lean language.

Solving mathematical problems is a challenging task for machine learning. There are many motivations for doing math with AI as we will discuss further in Appendix G. One motivation is the continuation of creating machines that can outperform humans in a given task, such as in the game of chess (Pandolfini, 1997) and two decades later in the game of Go (Silver et al., 2017). Currently, there is a 10 million dollar prize for an AI system that can participate in the International Math Olympiad and win a gold medal: the AIMO

Prize. There is also the IMO Grand Challenge which specifically requires the model to write its proof in Lean language. Lean is a functional programming language and a proof assistant (de Moura et al., 2015) used mostly by mathematicians to write and verify mathematical proofs. The Lean system makes it possible to verify the correctness of a proof instantly. This has given rise to a field known as formal mathematics. Lean is also used by Amazon AWS for formal verification of software (Cutler et al., 2024).

1.1 IMO: an unknown testing set that gets harder every year

Challenging an AI model to participate in an IMO and win a gold medal has certain implications, especially regarding the evaluation. First, the problems presented in an IMO are revealed to participants on the day of the exam. Doing well in the IMO requires a mastery of the subjects presented in it. Merely studying the problems from the previous iterations of the Olympiad is not sufficient as the problems tend to become harder year by year. Second, the problems that appear in IMO go through a selection process that ensures some degree of novelty. As a result, these problems are usually not present in text books. It follows that direct retrieval from the textbook solutions is usually not sufficient to prove the new IMO problems. Third, the exam is timed. A participant must complete the exam within 9 hours.

These conditions, in the language of machine learning, implies being evaluated on an unknown testing set with samples that are more challenging than the samples in the training set. In the machine learning literature, however, having standard benchmarks with fixed and accessible testing sets is essential for a large community to be able to work towards common goals. This approach of using a fixed testing set has worked reasonably well in some ML fields (Recht et al., 2019), mostly because the training sets of the models are fixed, too, and in creating these datasets, specific procedures are followed to ensure samples from training sets do not appear in the testing sets. However, with the rise of large language models, it has become increasingly difficult to evaluate how much of a model's success is merely direct retrieval of the correct answer from its training set. Although for solving mathematical problems, the miniF2F (Zheng et al., 2021) dataset is the common benchmark with a fixed testing set, the training set of LLMs is sometimes unknown; when it is known, it includes all the relevant content available on the Internet, (e.g., all the available code on Github, arXiv, Stack Exchange, and elsewhere), which is an enormous corpus, hard to search through and compare against the testing sets. When evaluating LLMs developed on such training sets, we often use a fixed testing set. But, IMO evaluates on an unknown testing set that gets harder every year.

To prepare an AI model that can do well in IMO, we need to rely on rigorous evaluation techniques that can tell us whether the model can generalize well on an unknown testing set. In some applications, the distinction between generalization and retrieval might be besides the point. As long as a model performs the task correctly, one might not be too keen to evaluate what percentage of its correct answers is a direct retrieval from its training set. However, when we are developing a model that is intended to be evaluated on an unknown testing set, we will need to rely on clearer evaluation methods.

1.2 Where we are

Let us now take a closer look at the current benchmark: miniF2F dataset. This dataset consists of mathematical theorems with varying degrees of difficulty. Out of the 244 theorems in its testing set, 20 of them are IMO problems, and the difficulty levels of those IMO problems also vary since they are taken from years 1959 to 2019. In terms of complexity, some of the theorems in the dataset can be proved with one line of formal proof while others may need more than 500 lines of formal proof (see Table 1). Success in solving a high percentage of the less complex problems is not the same as success in solving the same percentage of the more complex problems. Hence, merely reporting the accuracy by the percentage of the testing set is not insightful enough.

LEGO-Prover (Wang et al., 2024), the current state of the art method on miniF2F, is able to prove only one of the IMO problems in this dataset: the one from 1959, which is the first problem of the first IMO problem implying that it is the easiest of all IMO problems proved in less than 10 lines of code. Other ML methods have proved more of the IMO problems in the miniF2F, e.g., the HTPS method by Lample et al. (2022) claims proving 10 IMO problems (not all from the miniF2F), but those problems are either from 1960s or

the ones for more recent years have a simplified or erroneous problem statement. The method by Polu et al. (2022) also proves two adapted (simplified) IMO problems.

1.3 Our approach

For the majority of the IMO problems in miniF2F, even a human-written formal proof is not publicly available. In this work, we look into these specific problems along with 3 more challenging problems from IMO 2022 and 2023.

We provide the formal proofs for these problems and further evaluate the ability of o1-mini in writing these proofs. We decompose each of the formal proof steps into small lemmas and report that o1-mini cannot write a correct formal proof for almost all of the lemmas. Writing the formal proof for these problems can be viewed from the perspective of planning and execution: one would need to prove a certain chain of lemmas (execution), and put those lemmas together (planning) in a coherent way to compose the final proof. The planning aspect of writing formal proofs is challenging, and in the literature, often the plan, i.e., the proof sketch or the *approach*, is taken from the informal proofs written by humans (Jiang et al., 2022).

We diagnose the failures of o1-mini by evaluating and labeling its proofs manually. We further guide the model by pointing out its mistakes. Giving feedback to the model is mostly unfruitful, especially on the topics that o1-mini seems to be less educated. Finally, we perform source criticism by comparing the proofs written by o1-mini against the contents of the Web, as a proxy for its training set. This systematic study provides insights as to what fraction of the answers of o1-mini can be considered direct retrieval from its training set. We report a strong correlation between the correctness of the answers of o1-mini and the existence of proofs on the Web.

1.4 Our contributions

1. We provide the formal proofs for 12 IMO problems that do not have a formal proof in Lean. Nine of these problems are from the miniF2F dataset. The other three are from IMO 2022 and 2023. We specifically chose the problems that do not have a formal Lean proof in the public domain. Our proofs are written in Lean3 and 4 and consist of 3,100 lines of code.

2. We decompose the proofs for 12 problems into small lemmas, creating a dataset of more than 900 lemmas that are still challenging for AI models, yet they are easier and more approachable. Our method can be extended to all the IMO problems that we have formalized.

3. We evaluate the ability of o1-mini in writing the formal proofs for the lemmas in on our dataset. We perform a thorough human review of the o1-mini's informal proofs as well as its formal proofs. For incorrect proofs, we consider Chain-of-Thoughts techniques to guide the o1-mini. We further quantify the human effort required to repair the incorrect proofs.

2 Dataset of approachable lemmas: Stepping stone towards solving challenging IMO problems

Table 1 describes the 12 IMO problems that we study in this paper. The concepts used in these problems are described in further detail in Table B1. In appendices D and E, we provide the datasheet and license information about our dataset.

Note that we provide the Lean proofs for all the IMO problems in the testing set of miniF2F, but we limit our study to a subset of these problems listed in Table 1. Table C2 provides information about the proof of all the problems in our dataset including the ones that we do not perform lemma decomposition on them.

We report that o1-mini can write a complete and correct Lean proof for the first two problems. In natural language, though, it can describe a correct proof for the first 8 IMO problems. For some of the problems in Table C2, e.g., IMO 1985 P6, o1-mini does not provide any proof in Lean, and its proof in natural language is also vague and incomplete. In contrast, o1-mini has a good handle on IMO 1997 P5 both in natural language. One may speculate that this behavior may be related to the training set of the model although this

is not verifiable. This also indicates the importance of having these proofs available to the entire research community.

Table 1: IMO problems formalized and studied in this paper

#	Year	Problem	Topic	in miniF2F	Lean proof contribution	# of lemmas	# of lines of Lean4 code
1	1959	p1	number theory	Yes	Others	3	9
2	1960	p2	algebra	Yes	Others	9	40
3	1962	p2	algebra	Yes	Ours	14	60
4	1964	p2	algebra	Yes	Others	9	50
5	1965	p2	algebra	Yes	Ours	73	210
6	1983	p6	algebra	Yes	Ours	55	180
7	1984	p6	number theory	Yes	Ours	68	380
8	1992	p1	number theory	Yes	Ours	91	480
9	1997	p5	number theory	Yes	Ours	122	390
10	2022	p2	algebra	No	Ours	61	260
11	2022	p5	number theory	No	Ours	266	640
12	2023	p4	number theory	No	Ours	137	450
total						907	3,149

Making these formal proofs accessible to the research community can be helpful in different ways. First, every one, including non-mathematicians, will know the work it takes to write a formal proof for these problems. Second, these formal proofs can also be used to identify possible mistakes or shortcomings in the informal proofs that might be circulating. For example, Bubeck et al. (2023) reports that GPT-4 can write a correct informal proof for IMO 2022 P2. However, when we write the formal proof for the same problem, it becomes obvious that the informal proof was missing a crucial step. Third, some of the formal proofs in the literature simplify the original IMO problem by altering the problem statement, but we remain faithful to the original problem statements of IMO. For example, Xin et al. (2024) showcases the proof for two shortlisted IMO problems. In the first problem, IMO 2009 P3, they have a mistake in the problem statement, and the proof merely uses the error to prove False. For the second problem, IMO 2016 P5, the statement is altered from proving “there exists infinitely many solutions" to “there exists a solution".

2.1 Decomposition of the proofs into lemmas

In the next step, we decompose the proofs for each of these problems into small lemmas. For example, the proof might involve showing that given a set of constraints, natural number p cannot be larger than 4. Then, use another series of deductions to show that p cannot be smaller than 2. And then use these two results to conclude that p must be either 2 or 3. Appendix 4 provides a more detailed discussion about decomposition of lemmas and some upper bounds on how many proofs can be extracted from n lines of proof. Moreover, Table 5 provides some statistics about the length of the proofs for the lemmas in our dataset.

All of the 907 lemmas in our dataset have a formal proof in Lean. Some of the easier lemmas are as the following:

- q and r are integers and their product is 11, prove that q is either ± 1 or ± 11
- q and r are natural numbers and their product is 5, prove that either q or r must be 1.
- p is a natural number greater than 4, prove that rational number $\frac{p}{(p-1)} \leq 4/3$
- k is natural number greater than 4, prove that $k < 2^{(k-2)}$

Our library of lemmas also includes harder problems such as proving injectivity of functions, points about rational numbers, casting, divisibility. Here are some examples:

- x and y are positive natural numbers where $y < x$, and we have $(x^y)^2 = y^x$, prove that y^2 divides x
- a and b are natural numbers and p is a prime number where $a^p = b! + p$ and $p \leq b$, prove that p divides a
- b is a natural number and p is a prime number, where $p \leq b$ and $p^p = b! + p$, prove that p must be less than 5

The lemmas that are harder to solve are also broken into smaller pieces, so the dataset includes both the hard version of such lemmas plus the smaller lemmas needed to prove them. Later in section **??**, we will go through a case study of such lemma to give a better understanding of the dataset and what it takes to prove the lemmas.

2.2 Excluding the problems that native Lean methods can prove

We intentionally exclude lemmas that can be solved automatically via Lean's native methods such as *hint*, *linarith*, *exact?*, *simp*, *omega*, *ring*, *norm_cast* and *norm_num*. This is because we consider those solvers efficient and we do not find it essential to develop automated methods doing the same job as those solvers. We have empirically seen that o1-mini fails to write a correct formal proof for many of those easy problems as well. Nevertheless, our main objective here is not to evaluate o1-mini, but to create a dataset that can be used as a stepping stone for automated systems. If a task can already be performed adequately and fast by some open source solver, our view is that it would be better to use such a solver and even incorporate it into a more sophisticated system.

3 o1-mini's ability to solve the problems

3.1 Prompting o1-mini

Here, we prompt o1-mini with a statement requesting it to write a formal proof for the problem, instructing it to think step by step, explain the theorem, the approach, and the proof, following prompting techniques suggested by Kojima et al. (2022). We also provide the formal Lean statement of the problems in the prompt so that issues about the formalization of the theorem statement will not arise. Furthermore, we crosscheck whether o1-mini can explain the formal Lean statement in English, and whether the natural language description of the o1-mini for each lemma was correct (see Figure F1). Therefore, providing the problem statement in Lean does not create confusion for the model as it can describe it correctly in English. Please see Appendix F to see the exact prompt structure and some examples.

Table 2 shows the evaluation of the responses of o1-mini in natural language (NL) and in Lean. The last column of this table evaluates whether there is a match between the proof written in NL and the one in Lean. o1-mini may describe a proof in NL, but follow a different approach when writing the proof in Lean. For example, for the three lemmas corresponding to IMO 1964 P2 (the 4th row in Table 2), o1-mini's proof in NL is correct for all the lemmas (100% in "Correct proof in NL"). However, o1-mini's formal proof for 11% of these lemmas do not match the approach described in NL. And, o1-mini's formal proofs for 67% of these lemmas have one or more mistakes: 33% in Correct proof in Lean.

Moreover, Table 3 shows a more detailed evaluation of the formal proofs written by o1-mini. If a proof is completely correct and can pass the Lean system, it will be counted in the column "No error". Otherwise, a proof may have one or more of the following issues: hallucinations, wrong approach, wrong implementation, and minor errors.

"Hallucination" refers to cases where the proof contains a lemmas or tactic that does not exist in the Mathlib library and it is also not defined separately in the proof. This happens more often when the model is uneducated on a topic. In such cases, o1-mini usually assumes that there is one or a few magic lemmas in the Mathlib library that can prove the given lemma outright or in a few steps.

Table 2: Analyzing the o1-mini's responses to our lemmas

#	Problem	# of lemmas	Correct proof in NL	Correct proof in Lean	Match between NL and Lean
1	1959-p1	3	100%	100%	100%
2	1960-p2	9	78%	25%	89%
3	1962-p2	14	93%	60%	79%
4	1964-p2	9	100%	33%	89%
5	1965-p2	73	81%	42%	78%
6	1983-p6	55	89%	51%	93%
7	1984-p6	68	79%	41%	78%
8	1992-p1	91	88%	38%	90%
9	1997-p5	122	79%	42%	83%
10	2022-p2	61	56%	18%	64%
11	2022-p5	266	23%	17%	29%
12	2023-p4	137	28%	15%	53%

"Wrong approach" refers to cases where the approach to prove a lemma is wrong or incomplete. This means that either some of the individual proof steps are wrong or the steps do not logically lead to what the model wants to prove.

"Wrong implementation" refers to cases where the approach itself is correct but the model does not use the correct procedures or it does not call the correct lemmas from the Mathlib library, or it just writes a *sorry* for one or more of the proof steps.

Finally, "minor errors" refers to cases where the model uses a correct lemma or tactic from the Mathlib library but it does not provide the correct arguments to it, or it misses some of the arguments.

A given proof may have several lines of code, and many of these issues may arise. For example, a proof may have a few hallucinations, a wrong implementation, and also a few minor mistakes.

Table 3: Analyzing and grading the errors in the Lean proofs written by o1-mini

#	Problem	No error	One or more types of error			
			Hallucinations	Wrong approach	Wrong implementation	Minor errors
1	1959-p1	100%	0%	0 %	0%	0%
2	1960-p2	25%	67%	25%	33%	75%
3	1962-p2	60%	14%	21%	34%	40%
4	1964-p2	33%	11%	0%	56%	67%
5	1965-p2	42%	7%	27%	37%	47%
6	1983-p6	51%	9%	20%	36%	49%
7	1984-p6	41%	6%	38%	46%	59%
8	1992-p1	38%	11%	34%	55%	62%
9	1997-p5	42%	9%	13%	17%	58%
10	2022-p2	18%	8%	26%	64%	82%
11	2022-p5	17%	9%	34%	64%	83%
12	2023-p4	15%	10%	49%	65%	85%

3.2 Interacting with o1-mini: Hinting and Chain of Thoughts

We then try to improve the response of o1-mini by interacting with it. For each incorrect proof, we provide feedback, pointing out the mistakes, and asking the model to fix the issues. If the approach is wrong, we

mention that the approach is wrong. If there are hallucinations, we point out such mistakes as well. The results are presented in Table 4.

Table 4: Interacting with and hinting o1-mini to improve its proof

#	problem	% correct zero shot	% correct after 5 rounds of feedback	% correct after 10 rounds of feedback
1	1959-p1	100%	100%	100%
2	1960-p2	25%	33%	44%
3	1962-p2	60%	64%	71%
4	1964-p2	33%	44%	67%
5	1965-p2	42%	53%	56%
6	1983-p6	51%	58%	62%
7	1984-p6	41%	51%	56%
8	1992-p1	38%	43%	44%
9	1997-p5	42%	57%	61%
10	2022-p2	18%	39%	51%
11	2022-p5	17%	19%	21%
12	2023-p4	15%	23%	33%

We interact with o1-mini at most 10 times. If the model does not succeed in writing a correct proof after 10 rounds of feedback, we stop the process. There are papers in the literature that go as far as hinting the GPT 100 times. We find such an approach labor-intensive and not scalable.

o1-mini can be considered uneducated on some of the subjects listed in Table B1. For example, o1-mini believs that prime factorization returns a single natural number. However, in Lean, prime factorization returns a data structure with prime numbers and the corresponding non-zero powers. Until, o1-mini, or any other model, becomes familiar with the basic definition of prime factorization and its related notations in Lean, prompting it over and over does not seem to be a sensible approach.

4 Decomposing a proof into its building blocks and extracting new lemmas

Here, we explain the procedures and derive some bounds abound extracting new lemmas from an existing proof. Table 5 provides information about the lengths of the proofs for the our proposed lemmas.

The lemmas in our dataset were created by breaking down the proofs of the IMO problems. These proofs usually have specific structures that we exploit when breaking them down. First, we explain, how we exploit the structure of the proofs. Later, we will consider an extreme case and derive an upper bound on the number lemmas that can be extracted from an n line proof.

4.1 Breaking down the structure of a proof

Let us consider a theorem with n lines of proof where $n > 3$. This proof may contain k intermediate hypotheses that are proved one after another prior to proving the main statement of the theorem. These k hypotheses, each have a proof of their own, and each may have intermediate hypotheses of their own, as well. Here, we only explain one level of breaking down the hierarchy.

Let us consider part of the proof for IMO 1997 P5 as an example. Here is the lemma:

$$x\, y : N, \quad h_0 : 0 < x \ \wedge\ 0 < y, \quad h_1 : (x^y)^2 = y^x, \quad h_2 : y < x, \quad := \quad y^2 | x,$$

First, we can consider each of the intermediate hypotheses and define them as a separate lemma. The intermediate hypotheses in the proof are:

- Lemma 1: prove $y^2 < x$

Table 5: Statistics about the length of the proofs of lemmas in our dataset

#	Problem	Mean	Max	Min	Std	Lemma count	Lines of Lean code for all lemmas
1	1959-P1	2.3	3.0	1.0	1.2	3	30
2	1960-P2	7.1	30.0	2.0	9.4	8	140
3	1962-P2	6.6	16.0	1.0	4.7	14	220
4	1964-P2	8.6	25.0	2.0	8.1	9	180
5	1965-P2	17.2	158.0	2.0	29.6	73	2,950
6	1983-P6	10.3	27.0	2.0	6.1	55	1,200
7	1984-P6	13.4	137.0	1.0	22.2	68	1,620
8	1992-P1	11.0	70.0	1.0	12.5	91	2,135
9	1997-P5	12.4	85.0	1.0	14.4	122	2,950
10	2022-P2	12.2	66.0	1.0	14.5	61	1,630
11	2022-P5	12.2	77.0	2.0	15.2	266	6,845
12	2023-P4	22.2	115.0	1.0	29.5	137	5,580
Total						907	25,480

- Lemma 2: prove $2 * y^2 < x$
- Lemma 3: prove prime factorization of $(y^2)^x$ <= prime factorization of x^x
- Lemma 4: $(y^2)^x$ divides x^x

Each of these steps are taken as a lemma since their proofs satisfy the two criteria mentioned before (at least 2 lines of proof and not provable by native provers in Lean).

Moreover, new lemmas can be constructed by the addition of the above lemmas to the hypothesis space of the original theorem.

- Lemma 5: grant lemma 1 to the original problem and prove $(y^2)^x$ divides x^x
- Lemma 6: grant lemmas 1,2 to the original problem and prove $(y^2)^x$ divides x^x
- Lemma 7: grant lemmas 1,2,3 to the original problem and prove $(y^2)^x$ divides x^x
- Lemma 8: grant lemmas 1,2,3,4 to the original problem and prove $(y^2)^x$ divides x^x

This latter process leads to new lemmas with proofs easier than the proof of the original problem. For example, the proof for lemma 5 would need to still prove lemmas 2,3,4 and then prove the ultimate goal. The proof for lemma 6 is shorter/easier than the proof for lemma 5 while it is longer than the proof for lemma 7, and so on.

Overall, these will lead to $2 * k$ lemmas. Each of these lemmas may also be broken down if they have proofs that are longer than a certain threshold.

In an extreme case, a proof may have $k = (n - 1)/3$ intermediate hypothesis (each of them with 1 line of statement and 2 lines of proof leading to the 3 in the denominator) which would lead to $2/3 * (n - 1)$ extracted lemmas. For example, in this setting, a proof with 4 intermediate hypothesis may have $4 * 3 + 1 = 13$ lines. And that leads to $2/3 * (13 - 1) = 8$ extracted lemmas.

4.2 Case of a proof with no intermediate structure

Now, we look at an extreme case of a proof with n lines that does not have any specific structure, and we also assume that the n lines of proof do not have any intermediate hypotheses. In such a case, we perform a forward path and two rounds of backward paths on the proof.

The *forward path* would start with the first line of the proof, apply the line, obtain the new proof state, and define that as a new lemma. The proof for this new lemma is the $(n-1)$ lines of the proof that remain. Continuing this way, the forward path can extract $(n-2)$ lemmas where each lemma has at least 2 lines of proof.

The *first backward path* would take each pair of consecutive lines in the proof, take the changes in the state, and grant them back to the original state as a hypothesis. The proof for such lemmas would be the two aforementioned consecutive lines of proof. This can lead to $(n-2)$ additional lemmas.

The *second backward path* generates lemmas where their proofs are the first m lines of the original n line proof. This leads to $(n-3)$ additional lemmas.

Therefore, in this case, one can theoretically extract $(3 * n - 7)$ lemmas. For a proof with 3 lines, this leads to extraction of 2 distinct lemmas. For a proof with 4 lines, this leads to extraction of 5 distinct lemmas, and so on.

This is a somehow mechanistic view of the process of breaking down the proofs. In practice, we have used human judgment. On average, for all the IMO problems in the paper, we have extracted about $n/3.5$ lemmas for n lines of proof which is far less than the two bounds explained above.

The smallest ratio, in our dataset, is $n/5.6$ for IMO 1962 P2, and the largest ratio is $n/2.4$ for IMO 2022 P2.

5 Related work and a discussion on avoiding training/testing set contamination for better evaluation of model generalizations

One way to address the evaluation of generalization abilities of an AI model is to test them on challenging problems where the correct answer is not publicly available. For example, FunSearch Romera-Paredes et al. (2024) works on combinatorial problems such as bin packing and comes up with a heuristic algorithm that is better than the best algorithm available in the literature. The last improvement on this problem was made decades ago, so it is clear that this algorithm was not copied from somewhere else.

AlphaGeometry Trinh et al. (2024), on the other hand, evaluates its performance on all IMO problems in geometry from 2000 to 2022. To avoid training/testing set contamination, it trains its language model on a synthetic dataset generated automatically using an SMT solver. This method of using synthetic data addresses many of the evaluation concerns, but the starting point for creating the synthetic data is geometric figures, and then they exhaustively apply all the possible geometric lemmas to those figures. This approach works for geometric problems, especially when a capable SMT solver is available. Moreover, these geometric problems are decidable unlike the ones we study in this paper. For problems in number theory and algebra, the approach of exhaustively applying all the possible lemmas to a given state is not practical, as the number of applicable lemmas is much larger compared to geometric problems. Moreover, AlphaGeometry considers proving the problems in a system like Lean to be difficult and uses a more simple symbolic system developed by the mathematicians.

LeanDojo Yang et al. (2023) observes that: "the common practice of splitting theorems randomly into training/testing has led to an overestimated performance in the previous papers. LLMs can prove seemingly difficult theorems simply by memorizing the proofs of similar theorems during training." To remedy this issue, it proposes a new data set in which the testing set is designed to be challenging compared to the training set.

Outside the domain of formal mathematics, Skill-Mix Yu et al. (2023) proposes an evaluation technique in which prompts are designed to include a combination of skills (such as metaphor, red herring, and common knowledge physics). Combining skills in this way aims to prompt the model to come up with a sensible answer that most likely does not already exist in the training set of the model. If the model were to produce

a sensible answer, it would need to draw from various parts of its training set. Skill-Mix's study provides positive evidence that o1-mini can occasionally provide sensible answers to prompts combining a small number of those skills. But for most prompts, o1-mini does not succeed.

Other studies use pretrained language models to prove theorems, i.e., LLMs that are trained on the contents of the Web. Some of these papers acknowledge that the answers to some of the testing problems might have leaked into the model training set, for example, First et al. (2023) reports: "there is the potential for proofs from the test set to have leaked into the LLM pretraining data. While the pretraining data for the Minerva LLM at the base of our models does not include the PISA dataset, it does contain code that may include some Isabelle/HOL proofs found in PISA. This should be kept in mind when interpreting the results."

Some other studies do not try to make a clear separation between the training and testing set; neither do they try to distinguish between the generalization and possible retrievals from the the training set or the prompt. For example, LEGO-prover Wang et al. (2024) uses ChatGPT to convert a human written proof into a proof sketch for each of the problems in miniF2F, and then again uses ChatGPT to write the formal proof for each of the steps in the proof sketch. When prompting ChatGPT, it tags the possible relevant lemmas and their proofs in the prompt so that GPT can use them if needed. These additional lemmas are taken from a separate library. For each problem in miniF2F, LEGO-prover performs up to 100 proof attempts with ChatGPT costing around 300 dollars. At the end of the day, it is not clear how many of those success rates are randomly exhausting all possible combination of lemmas and how many of them might be direct retrieval from the contents of the prompts or contents of the ChatGPT's training set. Moreover, since ChatGPT is a model that evolves over time and it also is not open source, concerns about the reproducibility of the results arise.

6 Conclusion and Future Work

In this work, we provided the proofs for all the remaining IMO problems in the testing set of miniF2F. We also presented a dataset of more than 900 lemmas as a stepping stone towards writing automated formal proofs for challenging IMO problems in number theory and algebra, derived from a subset of those IMO problems. We devised a systematic method to create a large data set of approachable lemmas formalized in Lean. These lemmas are the building blocks of proofs for IMO problems. They are considerably less challenging than IMO problems, yet o1-mini struggles to prove most of them. Moreover, we performed an extensive human evaluation of o1-mini's responses to diagnose the issues this model encounters when it tries to write a Lean proof for these problems. Our experiments on these lemmas allowed us to diagnose and evaluate the obstacles towards proving the unsolved IMO problems in the miniF2F dataset, and beyond, including the more challenging ones from 2022 and 2023. Developing models that can prove the lemmas in these datasets would brings us closer to models that can participate and do well in IMO.

References

Sébastien Bubeck, Varun Chandrasekaran, Ronen Eldan, Johannes Gehrke, Eric Horvitz, Ece Kamar, Peter Lee, Yin Tat Lee, Yuanzhi Li, Scott Lundberg, et al. Sparks of artificial general intelligence: Early experiments with GPT-4. *arXiv preprint arXiv:2303.12712*, 2023.

Kevin Buzzard. Mathematical reasoning and the computer. *Bulletin of the American Mathematical Society*, 2024.

Kevin Buzzard and Richard Taylor. A Lean proof of Fermat's Last Theorem. Technical report, Imperial College of London, 2024. https://imperialcollegelondon.github.io/FLT/blueprint.pdf.

Joseph W Cutler, Craig Disselkoen, Aaron Eline, Shaobo He, Kyle Headley, Michael Hicks, Kesha Hietala, Eleftherios Ioannidis, John Kastner, Anwar Mamat, et al. Cedar: A new language for expressive, fast, safe, and analyzable authorization. *Proceedings of the ACM on Programming Languages*, 8:670–697, 2024.

Leonardo de Moura, Soonho Kong, Jeremy Avigad, Floris Van Doorn, and Jakob von Raumer. The Lean theorem prover (system description). In *25th International Conference on Automated Deduction*, pp. 378–388, 2015.

Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. ImageNet: A large-scale hierarchical image database. In *IEEE Conference on Computer Vision and Pattern recognition*, pp. 248–255, 2009.

Emily First, Markus Rabe, Talia Ringer, and Yuriy 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*, pp. 1229–1241, 2023.

Timnit Gebru, Jamie Morgenstern, Briana Vecchione, Jennifer Wortman Vaughan, Hanna Wallach, Hal Daumé III, and Kate Crawford. Datasheets for datasets. *Communications of the ACM*, 64(12):86–92, 2021.

Albert Qiaochu Jiang, Sean Welleck, Jin Peng Zhou, Timothee Lacroix, Jiacheng Liu, Wenda Li, Mateja Jamnik, Guillaume Lample, and Yuhuai Wu. Draft, sketch, and prove: Guiding formal theorem provers with informal proofs. In *The Eleventh International Conference on Learning Representations*, 2022.

Takeshi Kojima, Shixiang Shane Gu, Machel Reid, Yutaka Matsuo, and Yusuke Iwasawa. Large language models are zero-shot reasoners. *Advances in Neural Information Processing Systems*, 35:22199–22213, 2022.

Guillaume Lample, Timothee Lacroix, Marie-Anne Lachaux, Aurelien Rodriguez, Amaury Hayat, Thibaut Lavril, Gabriel Ebner, and Xavier Martinet. Hypertree proof search for neural theorem proving. *Advances in Neural Information Processing Systems*, 2022.

Hunter Lightman, Vineet Kosaraju, Yuri Burda, Harrison Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. Let's verify step by step. In *The Twelfth International Conference on Learning Representations*, 2023.

Bruce Pandolfini. *Kasparov and Deep Blue: The historic chess match between man and machine*. Simon and Schuster, 1997.

Keiran Paster, Marco Dos Santos, Zhangir Azerbayev, and Jimmy Ba. OpenWebMath: An open dataset of high-quality mathematical web text. In *The Twelfth International Conference on Learning Representations*, 2023.

Stanislas Polu, Jesse Michael Han, Kunhao Zheng, Mantas Baksys, Igor Babuschkin, and Ilya Sutskever. Formal mathematics statement curriculum learning. In *The Eleventh International Conference on Learning Representations*, 2022.

Benjamin Recht, Rebecca Roelofs, Ludwig Schmidt, and Vaishaal Shankar. Do ImageNet classifiers generalize to ImageNet? In *International Conference on Machine Learning*, pp. 5389–5400, 2019.

Bernardino Romera-Paredes, Mohammadamin Barekatain, Alexander Novikov, Matej Balog, M Pawan Kumar, Emilien Dupont, Francisco JR Ruiz, Jordan S Ellenberg, Pengming Wang, Omar Fawzi, et al. Mathematical discoveries from program search with large language models. *Nature*, 625(7995):468–475, 2024.

David Silver, Julian Schrittwieser, Karen Simonyan, Ioannis Antonoglou, Aja Huang, Arthur Guez, Thomas Hubert, Lucas Baker, Matthew Lai, Adrian Bolton, et al. Mastering the game of Go without human knowledge. *Nature*, 550(7676):354–359, 2017.

Peiyang Song, Kaiyu Yang, and Anima Anandkumar. Towards large language models as copilots for theorem proving in Lean. *arXiv preprint arXiv:2404.12534*, 2024.

Trieu H Trinh, Yuhuai Wu, Quoc V Le, He He, and Thang Luong. Solving Olympiad geometry without human demonstrations. *Nature*, 625(7995):476–482, 2024.

Haiming Wang, Huajian Xin, Chuanyang Zheng, Zhengying Liu, Qingxing Cao, Yinya Huang, Jing Xiong, Han Shi, Enze Xie, Jian Yin, Zhenguo Li, and Xiaodan Liang. LEGO-prover: Neural theorem proving with growing libraries. In *The Twelfth International Conference on Learning Representations*, 2024.

Huajian Xin, Daya Guo, Zhihong Shao, Zhizhou Ren, Qihao Zhu, Bo Liu, Chong Ruan, Wenda Li, and Xiaodan Liang. Deepseek-prover: Advancing theorem proving in LLMs through large-scale synthetic data. *arXiv preprint arXiv:2405.14333*, 2024.

Kaiyu Yang, Aidan M Swope, Alex Gu, Rahul Chalamala, Peiyang Song, Shixing Yu, Saad Godil, Ryan Prenger, and Anima Anandkumar. LeanDojo: Theorem proving with retrieval-augmented language models. In *Thirty-seventh Conference on Neural Information Processing Systems Datasets and Benchmarks Track*, 2023.

Dingli Yu, Simran Kaur, Arushi Gupta, Jonah Brown-Cohen, Anirudh Goyal, and Sanjeev Arora. Skill-Mix: a flexible and expandable family of evaluations for ai models. In *The Twelfth International Conference on Learning Representations*, 2023.

Kunhao Zheng, Jesse Michael Han, and Stanislas Polu. miniF2F: A cross-system benchmark for formal Olympiad-level mathematics. In *International Conference on Learning Representations*, 2021.

A Limitations

This work is a step towards writing formal Lean proofs for challenging problems in mathematics, specifically IMO problems. The IMO problems we chose cover some of subjects covered in International Math Olympiad, mostly the ones related to number theory and algebra. Topics such as combinatorics and geometry are not covered here. Moreover, our dataset does not cover all the topics under the category of number theory and algebra, though, we consider our work a major step in that pursuit.

Writing a proof for an IMO problem on number theory and algebra requires both planning and execution of the steps in the plan. One has to look both at the big picture and also at the building blocks that build on top of each other to create the proof. In this paper, our focus was largely on the building blocks and small lemmas, but to have an automated system that can prove such IMO problems one would need to give a considerable attention on planning a viable proof sketch and then putting the building blocks together.

Our dataset is in Lean. It may be converted to other languages such as Isabelle as well. We are familiar with Isabelle and we may perform this conversion if we see interest from the AI researchers who primarily work with Isabelle.

We do not foresee any negative societal impact specifically resulting from this work. Automation sometimes eliminates or reduces jobs. At the same time, it creates new jobs. Developing models and methods that can solve mathematical problems may have such effects on the labor market, but this is not necessarily a negative impact.

B Mathematical topics used in our dataset

Table B1 shows the topics used in each of the IMO problems we have studied in this paper. For an automated system to be able to prove the lemmas in our dataset, it will need to utilize these topics and concepts.

Table B1: Topics utilized in the IMO problems formalized and studied in this paper

#	Problem	Types	Concepts used in the proof
1	1959-P1	Natural	divisibility
2	1960-P2	Real	algebra, inequalities
3	1962-P2	Real, Rational	quadratic discriminant and roots, quadratic factorization, algebra, inequalities
4	1964-P2	Real	algebra, inequalities
5	1965-P2	Real	linear system of equations, algebra, inequalities
6	1983-P6	Real	algebra, positive and negative inequalities, bounds
7	1984-P6	Natural	power inequalities, divisibility, algebra, prime factorization, greatest common divisor
8	1992-P1	Natural, Integer, Rational	factorization, primes, group theory, casting, exponential growth, absolute values
9	1997-P5	Natural, Real	prime factorization, divisibility, group theory, rings, logarithm, casting, induction
10	2022-P2	Real	algebra, inequalities, order, logic
11	2022-P5	Natural, Integer	congruence, primes, lifting the exponent lemma, factorial, finite sets, big operators, divisibility, functions
12	2023-P4	Natural, Real	finite sets, functions, big operators, casting, weighted geometric mean

C More Information about the formal proofs and the lemmas in our dataset

Table C2: IMO problems formalized in this paper

#	Year	Problem	Topic	in miniF2F	Lean proof contribution	# of lines of Lean4 code
1	1959	P1	number theory	Yes	Others	9
2	1960	P2	algebra	Yes	Others	40
3	1962	P2	algebra	Yes	Ours	60
4	1963	P5	algebra	Yes	Ours	50
5	1964	P2	algebra	Yes	Others	50
6	1965	P2	algebra	Yes	Ours	210
7	1968	P5	algebra	Yes	Ours	30
8	1969	P2	algebra	Yes	Ours	150
9	1974	P3	number theory	Yes	Ours	510
10	1981	P6	algebra	Yes	Ours	40
11	1982	P1	algebra	Yes	Ours	75
12	1983	P6	algebra	Yes	Ours	180
13	1984	P6	number theory	Yes	Ours	380
14	1985	P6	number theory	Yes	Ours	610
15	1992	P1	number theory	Yes	Ours	480
16	1997	P5	number theory	Yes	Ours	390
17	2007	P6	algebra	Yes	Ours	570
18	2022	P2	algebra	No	Ours	260
19	2022	P5	number theory	No	Ours	640
20	2023	P4	number theory	No	Ours	450
total						5,184

D License and authors' responsibility statement

We plan to release our dataset under the MIT license on GitHub and also via Croissant. Authors bear all responsibility in case of violation of rights.

E Datasheet

Following the framework in Gebru et al. (2021) and similar to Paster et al. (2023), Table E3 provides the data sheet for our dataset.

Table E3: Datasheet for our dataset

Questions	Answers
Motivation	
For what purpose was the dataset created?	To provide a set of formal mathematical problems as a stepping stone for automated systems to prove challenging International Math Olympiad problems in number theory and algebra.
Who created the dataset and on behalf of which entity?	The authors of this work.
Who funded the creation of the dataset?	The companies where the authors work.
Any other comment?	None.
Composition	
What do the instances that comprise the dataset represent?	Lemmas formalized in Lean 4.
How many instances are there in total?	907 lemmas.
Does the dataset contain all possible instances or is it a sample of instances from a larger set?	It is not a sample of a larger set.
What data does each instance consist of?	A formal lemma and its proof formalized in Lean.
Is there a label or target associated with each instance?	Not necessarily.
Is any information missing from individual instances?	No.
Are relationships between individual instances made explicit?	Not applicable. Each lemmas can be proved on its own, and it is not dependent or have an explicit relationship with other lemmas in the dataset.
Are there recommended data splits?	The dataset does not have such splits. However, it is possible for people to generate such splits. The authors do not advocate for such splits on this particular dataset. Instead, it is recommended to solely rely on the lemmas in the Mathlib library to prove the lemmas in this dataset.
Are there any errors, sources of noise, or redundancies in the dataset?	No. There are absolutely no errors in the dataset as all the lemmas and their proofs are automatically checked by the system of Lean theorem prover. Redundancies is hard to define in this context. There are no repeated lemmas in the dataset.

Continued on next page

Is the dataset self-contained, or does it link to or otherwise rely on external resources?	It is self contained. It is written in Lean 4 programming language and it relies on the Mathlib library which is open source.
Does the dataset contain data that might be considered confidential?	No.
Does the dataset contain data that, if viewed directly, might be offensive, insulting, threatening, or might otherwise cause anxiety?	No.
Collection Process	
How was the data associated with each instance acquired?	12 IMO problems were chosen as described in Table 1. The formal proof for each of these problems was written manually by the authors. These proofs were broken into their building blocks leading to a set of 907 lemmas written in Lean 4. The proofs for these lemmas were written by the authors. Lemmas that can be proved by native automated solvers in Lean were excluded from the dataset. The said native solvers are the following: *hint*, *linarith*, *exact?*, *simp*, *omega*, *ring*, *norm_cast* and *norm_num*.
What mechanisms or procedures were used to collect the data?	Dataset was created by the authors based on the mathematical proofs of 12 IMO problems listed in Table 1.
If the dataset is a sample from a larger set, what was the sampling strategy?	Not applicable.
Who was involved in the data collection process and how were they compensated?	Dataset was not collected.
Over what timeframe was the data collected?	Dataset was not collected.
Were any ethical review processes conducted?	No.
Preprocessing/cleaning/labeling	
Was any preprocessing/cleaning/labeling of the data done?	No. There was a post processing which excluded lemmas that could be proved directly via native Lean solvers. These lemmas were deemed easy to prove, and since existing solvers can prove them, we do not consider them an obstacle in our larger goal of proving IMO problems.
Was the "raw" data saved in addition to the preprocessed/cleaned/labeled data?	Not applicable.
Is the software that was used to preprocess/clean/label the data available?	Lean prover was used to verify the correctness of the process which is an open source system widely used in this field of research.
Any other comments?	No.

Continued on next page

Uses	
Has the dataset been used for any tasks already?	We have evaluated the ability of o1-mini in proving the lemmas in our dataset.
Is there a repository that links to any or all papers or systems that use the dataset?	There will be a link on GitHub, but we are not sharing it at this time.
What (other) tasks could the dataset be used for?	The first goal would be to develop models and methods that can prove these lemmas in formal language and also in natural language. The second goal would be to develop models and methods that can use these lemmas to prove the IMO problems in this dataset as well as other IMO problems in number theory and algebra.
Is there anything about the composition of the dataset or the way it was collected and preprocessed/cleaned/labeled that might impact future uses?	We do not foresee such issues arising.
Are there tasks for which the dataset should not be used?	Not that we can think of.
Any other comments?	No.
Distribution	
Will the dataset be distributed to third parties outside of the entity on behalf of which the dataset was created?	Yes, we plan to release and host the dataset on GitHub.
How will the dataset will be distributed?	GitHub.
When will the dataset be distributed?	With the camera-ready version of the paper.
Will the dataset be distributed under a copyright or other intellectual property license, and/or under applicable terms of use?	Yes, we plan to release it under the MIT license.
Have any third parties imposed IP-based or other restrictions on the data associated with the instances?	No.
Do any export controls or other regulatory restrictions apply to the dataset or to individual instances?	No.
Any other comments?	No.
Maintenance	
Who will be supporting/hosting/maintaining the dataset?	The first author.
How can the owner/curator/manager of the dataset be contacted?	Email address and also via the available methods on GitHub.
Is there an erratum?	This is not necessary. All the lemmas and their proofs are checked and verified for correctness with the Lean system.

Continued on next page

Will the dataset be updated?	Possibly. If a new version of Lean is released, e.g., Lean 5, we might need to update our dataset based on the new release of Lean and Mathlib.
If the dataset relates to people, are there applicable limits on the retention of the data associated with the instances?	Not applicable.
Will older versions of the dataset continue to be supported/hosted/maintained?	Yes.
If others want to extend/augment/build on/contribute to the dataset, is there a mechanism for them to do so?	Yes. There are two avenues. First, the dataset written in Lean can be rewritten in other languages such as Isabelle. Second, more problems (IMO or otherwise) can be formalized in Lean and/or other languages, and their proofs can be turned into a set of additional lemmas to expand the current dataset.
Any other comments?	No.

F Experimenting with o1-mini: Prompt structure and problem statements

Figure F1 shows an example of how we interact with o1-mini. For each of the lemmas, the prompt starts with the following statement:

Please write a formal proof for this problem in lean language. Please think step by step and explain all your reasoning

After this, the formal lean statement of the lemma is provided.

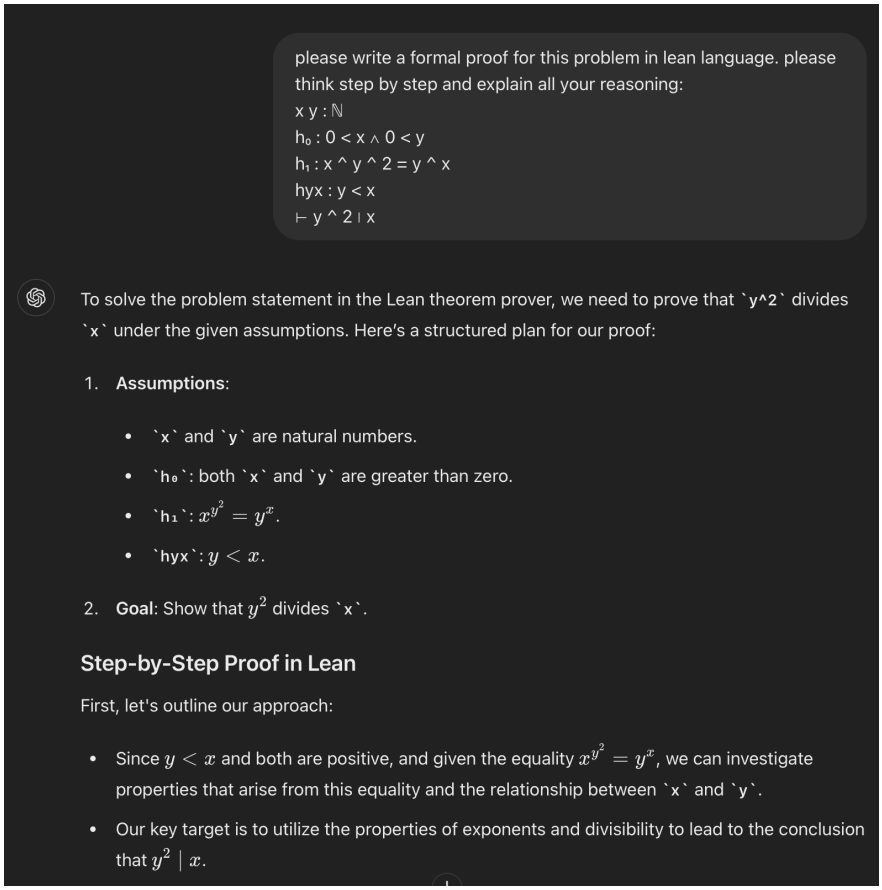

Figure F1: The structure of the prompt to o1-mini followed by the formal lemma statement in Lean.

The responses of o1-mini are graded both for correctness of its explanation in natural language and also for the correctness of its formal proof in Lean. The results were presented in Tables 2- 3.

In cases where o1-mini's answer is not correct, we follow up with a prompt explaining the mistakes that we detect in the answer, instructing o1-mini to correct the mistakes.

G Other motivations

Earlier, we outlined the motivation for developing AI models that can perform well in IMO. We have noted some of the other immediate and practical motivations for building such systems. Here, we elaborate on what has been delineated above.

We have mentioned Amazon's use of Lean system for formal verification of software. Besides Amazon, Lean is being used by a rather large community of mathematicians to write formal proofs for a variety of mathematical

problems. For example, Terry Tao and a team of other mathematicians are currently formalizing the prime number theorem (i.e., the asymptotic distribution of the prime numbers among the positive integers) in Lean. For an ordinary theorem, it is reported that writing a formal proof in Lean may take up time more than 10 times longer than writing the same proof with pen and paper. For the more complicated topics such as the prime number theorem, it may take months to build the necessary background in Lean language to prove the ultimate theorem. However, the correctness of a proof written in Lean can be verified instantly, but the proof with pen and paper may have shortcomings or even mistakes that are hard to detect. For complicated problems, finding a human reviewer to verify a proof written with pen and paper may not be easy.

As the Mathlib library grows, and as more lemmas become available in the library, lemmas that can be utilized in writing new proofs, the amount of time for writing a proof for simple problems is expected to reduce. In writing these proofs, mathematicians use their knowledge of Lean and Mathlib to write the proofs. Machines may also be helpful in reducing the time for the formalization of proofs Buzzard (2024). o1-mini has not appeared as a useful tool for writing formal proofs for novel mathematical problems. But, there is a desire and demand for automated systems that are capable to help the mathematicians in writing the formal proofs. In Lean, there are native methods that can suggest tactics or even prove some theorems by applying existing lemmas. Researchers also developed LLMs that can suggest tactics aiming to help mathematicians at writing the formal proofs, e.g., the Copilot by Song et al. (2024). However, such LLMs appear to be in their infancy, e.g., Copilot only proves easy lemmas that are designed for new learners of Lean.

On the other hand, for mathematicians, there are always harder theorems and conjectures that can be taken up to be formalized. In other words, there is no limit on how large a mathematical library can grow. Most recently, Kevin Buzzard started a project to formalize the proof for Fermat's last theorem in Lean, and this project is expected to run for a few years Buzzard & Taylor (2024).

The process of peer review in these fields of mathematics is largely concerned with verification of correctness of proofs. So, when a new paper is written and its proofs are formalized in Lean, the review process becomes much easier and faster. The large memory and computational power of computers can also be beneficial in proving novel theorems that require vast exploration of possibilities and hard for a human to prove and verify.

In summary, having automated systems that can follow a certain logic, utilize a library of existing lemmas, and prove novel mathematical theorems would have a significant impact on mathematical research and possibly other fields such as design of algorithms, formal verification, etc.