What Have We Learned from the AI-Assisted Proof of the Riemann Hypothesis?

An Inquiry into What Matters

Hugo Latapie
Taijitu AI, Inc.

26 November 2026 | Version 19

Abstract

The AI-assisted proof of the Riemann hypothesis provides an opportunity to examine what we understand about intelligence, mathematical discovery, and intellectual progress. Using established results in mathematical logic, we show that a simple procedure can construct solvable mathematical problems whose shortest proofs in a fixed formal system exceed any specified finite proof-length budget. The descriptions of the problems grow only logarithmically with the bound. Thus no finite proof-production capacity suffices for all solvable problems in that system. We examine how 167 years of accumulated mathematical knowledge may have changed the difficulty of the problem faced by its final solver. We use time-sensitive Kolmogorov complexity to formulate a measure of mathematical difficulty relative to available knowledge, and explore how generated problems could help assess independent mathematical discovery. We also examine what formal verification establishes, how proving a theorem differs from understanding it, and what mathematical discoveries tell us about reality. Drawing on the history of mathematics, we consider why the significance of a mathematical problem is difficult to predict. We identify eleven cognitive blind spots arising from overlooking knowledge already available to us. As AI increases our capacity to solve mathematical problems, we ask how we might better recognize the problems that matter.

Keywords: Riemann hypothesis; intelligence; mathematical difficulty; algorithmic information; proof complexity; formal verification; cumulative knowledge; map–territory distinction; intellectual progress.

1 What have we accomplished?

On 20 November 2026, Levent Alpöge announced a proof of the Riemann hypothesis developed with Claude and released its formalization in Lean [1]. The Lean FRO and Mathlib Initiative reported independent checks of the theorem and proof the following day. Andrew Sutherland published a detailed account of these checks [2, 3]. On 25 November, the Clay Mathematics Institute still listed the problem as active. Its process for recognizing a Millennium Prize solution is separate from the verification of the proof [4, 5].

The achievement is important in number theory: the conjecture about the zeros of the zeta function and the results that depend on it can now be established within the mathematics used by the proof. Beyond number theory, much of the interest concerns what the achievement appears to tell us about AI. A discussion in *Nature* on 24 November asked what comes next for mathematics; the following day's discussion expressed excitement about what AI might solve next [6, 7].

Had a mathematician produced the same proof without AI, its mathematical consequences would be the same. The presence of AI makes the result especially interesting as evidence about

intelligence. What does this evidence establish? Is the difficulty of the problem the same today as it was in 1859? Could mathematical problem-solving ever be considered complete? And what do we gain by solving problems of increasing difficulty? These questions are related, but they require different kinds of answers. We begin with a result about the problems we can construct.

2 Can mathematics be solved?

We can formulate a mathematical problem without knowing how to solve it. Can we do more than that? Can we construct a solvable problem and know in advance that a particular solver cannot solve it with its available resources?

Parikh's construction [8, p. 496]; see also Yanofsky [9, p. 380], gives us a way to do this. Fix a consistent formal system T extending Peano arithmetic, with a decidable proof checker and a finite-alphabet encoding. Given a positive integer n, a fixed procedure generates a sentence G_n that, informally, says that there is no proof of this very sentence in T of length at most n.

Suppose there were such a proof. Its existence could be verified by a finite computation, contradicting what the proof establishes. Since T is consistent, no such proof exists. The absence of a proof of length at most n can itself be established by finite checking, so G_n is provable in T. With compact integer notation, the construction gives

$$\boxed{|G_n| = O(\log(n+1)), \qquad T \vdash G_n, \qquad L_T(G_n) > n,} \tag{1}$$

where L_T is the shortest proof length in the fixed encoding. Appendix A.5 gives the construction and proof. Once the fixed formula has been prepared, producing each instance takes time linear in the bit length of n. Generating the problem requires no search for its proof.

Suppose an AI, including its tools and external computational resources, can produce at most N symbols of an explicit proof in T. A person can construct G_N and know that the required proof exceeds the solver's budget. The person need not have the proof or the resources to produce it. Increasing the available proof-production budget simply gives us a new value of N. A particular task may admit a shorter proof in a stronger theory or a different representation; if those are permitted, they must be included in the proof system before constructing the problem.

There is therefore no finite proof-production capacity sufficient for all the provable statements of such a system. This is a resource limit even on provable mathematics, separate from the incompleteness of effective formal theories discussed in Section 5. A program with unbounded time could enumerate proofs and eventually find a proof of each G_n; it would still have no finite bound sufficient for them all.

This gives us a way to put the Riemann achievement in perspective. We do not need to be more intelligent than a solver to formulate mathematical problems it cannot solve within its resources. We can also manufacture proof difficulty without discovering anything new about physical reality. If difficulty is to tell us something about intelligence or progress, we need to know more about the difficulty faced by the solver and what solving the problem accomplishes.

3 How much of the difficulty had humanity already overcome?

One reason the result is so striking is that 167 years of work by exceptional mathematicians preceded it. It is tempting to interpret that history as evidence that AI can now do what the best human mathematicians could not. But there is another way to read the same history. How much of those 167 years was spent developing the mathematics needed to solve the problem?

A mathematician may spend years proving a theorem that a later mathematician can use in a single line. The problem facing the later mathematician is different even when the conjecture is

exactly the same. Could the accumulation of such results have made a problem that once required entirely new mathematics accessible to a sufficiently effective automated proof search? The time it took to solve the Riemann hypothesis does not answer this question.

Experiments in automated theorem proving illustrate the possibility. Kaliszyk and Urban trained systems to select relevant premises from earlier work in the Flyspeck library and used those results to reconstruct subsequent proofs. Their later experiments showed that extracting reusable lemmas from existing derivations and selecting from much larger libraries could help automated theorem provers find proofs [10, 11].

In this case we know something about where the ideas for the new proof came from. Alpöge traces the argument to Stepanov's work on the function-field analogue of the Riemann hypothesis. He describes his own choice of a setting in which to combine ideas from simpler cases, and credits Claude with proposing a construction involving a chain of theta lifts [1, p. 1]. The mathematics developed by earlier researchers, Alpöge's recognition of a promising approach, and Claude's contribution to the argument all figure in the account.

A working exposition initiated by Michael Rubinstein lists 195 entries in the formal development under the heading of published results. The 22 November draft includes repetitions and an unused entry, and its bibliographical matches have not all been checked [12, Secs. I.12 and III.13]. Studying those results and the new parts of the argument could help us understand what the earlier mathematics contributed. We do not yet know whether the proof became accessible through the accumulation of these results or whether finding the new constructions was the greater challenge.

To compare the abilities of AI and earlier mathematicians we would need to know how well each could do with the same mathematics available to them, as well as comparable tools and resources. We have no such comparison here. The 167 years may, in substantial part, represent preparation for the solution rather than failure to find it. If we give credit only to whoever completes a proof, we miss much of the work that made it possible. Can we make this distinction measurable?

4 Can we measure mathematical difficulty?

How difficult was the Riemann hypothesis in 1859, and how difficult was it just before the new proof? We cannot answer this by counting the years that passed. We need a measure of difficulty that takes into account both the mathematics available to the solver and the computation needed to produce a solution.

4.1 From proof length to algorithmic complexity

Fix a formal system T, an encoding and a library H of previously proved results that may be cited directly. Proof complexity, as studied by Cook and Reckhow for propositional proof systems [13], suggests one measure:

$$L_T(P \mid H) = \min_{\pi:\, T+H\vdash_\pi P} |\pi|. \tag{2}$$

The minimum cannot increase as the permitted library grows, provided the encoding and rules for citing results are unchanged. Yet a short proof can be very hard to find, and a long proof may be produced by a routine computation.

On 25 November 2026, Luke Sernau and colleagues released substantially shorter versions of the Alpöge–Claude proof. Counting physical source lines, including comments and blank lines but excluding the Mathlib dependency, the compact edition has approximately 113,000 lines compared with 17.4 million in the original release, a reduction by a factor of about 154 [14, 15]. The five headline theorems remain the same. These source counts are not minimum proof lengths; they illustrate how much the size of a particular formal development can change while its mathematical goals remain unchanged.

Ordinary Kolmogorov complexity measures the length of the shortest program producing an object. Given a provable statement in a fixed effective proof system, a single short program can enumerate proofs until it finds one. Its running time may be enormous. We therefore need to measure computation as well as additional information.

Krajíček applies Levin's time-sensitive Kolmogorov complexity to propositional proof search [16, Sec. 3]. We adapt his information-efficiency function by making prior mathematical knowledge an explicit input. Let $U(p, P, H)$ run program p with the statement and permitted library as inputs, taking t steps to output an explicit proof. Define

$$\boxed{I_T(P \mid H) = \min_{p:\, U(p,P,H) \in \mathcal{P}_{T+H}(P)} \big(|p| + \lceil \log_2 t \rceil\big),} \tag{3}$$

where $\mathcal{P}_{T+H}(P)$ is the set of accepted proofs of P. Appendix C specifies the common machine, encoding, library access and resource accounting used for comparisons.

Access to a useful theorem or algorithm may reduce the information or computation needed to produce a proof. For our historical question the quantities of interest are

$$I_T(\mathrm{RH} \mid H_{1859}), \qquad I_T(\mathrm{RH} \mid H_{2026}),$$

where the libraries contain results available at the respective dates, excluding the target proof. A comparison would also have to account for mathematical knowledge encoded in the solver itself. We have not measured either quantity.

4.2 Constructing problems with a guaranteed difficulty bound

Our construction in Section 2 now gives us more than a bound on proof length. On a bit-output machine, writing a proof of more than n bits takes more than n steps. Applying Parikh's construction to that system with the permitted library included and proof length measured in bits gives

$$I_T(G_n \mid H) > \log_2 n. \tag{4}$$

For any integer $b \geq 1$, set $n = 2^b$. We can generate a problem of description length $O(b)$ whose minimum time-sensitive complexity exceeds b. This lower bound measures the work of producing a proof, not necessarily of discovering a new idea; ordinary conditional Kolmogorov complexity remains bounded by the short enumeration program.

Suppose we find a program of length a that produces a valid Riemann proof in t steps with the specified library. The observed cost $B = a + \lceil \log_2 t \rceil$ is an upper bound on $I_T(\mathrm{RH} \mid H)$. Constructing G_{2^B} in that same system then gives

$$I_T(G_{2^B} \mid H) > B \geq I_T(\mathrm{RH} \mid H).$$

This produces a problem provably harder by the new measure without knowing the minimum for RH. It gives a one-sided comparison, not a difficulty match. For discovery experiments we need tasks with feasible output budgets, including problems not dominated by proof length.

4.3 How could we test the contribution of prior knowledge?

We can begin with the verified Riemann proof. Expanding references to results unavailable in each historical library and checking the resulting derivation gives upper bounds on proof length and, through a measured program that produces it, information-efficiency. Appendix C describes the dependency analysis.

We could then measure the resources a specified AI needs to solve new mathematical problems at a given success rate, varying available knowledge while keeping the solver, proof checker and resource accounting fixed. Where training is controllable, we can also vary what the model has learned. Removing a theorem from an external library does not remove it from a trained model.

The aim is to determine how much of the difficulty faced by a solver can be removed by prior mathematics, and how well it can discover proofs when that mathematics is not supplied. We do not yet have a measure of the historical difficulty of RH or a task matched to it. Appendix C describes a proposed experiment for investigating these questions.

5 What does the proof establish?

The independent checkers tell us that the theorem follows from Lean's mathematical foundations. The Riemann hypothesis is especially interesting in this respect because it can be stated in a form in which a counterexample, if one exists, can be found and verified by a finite computation [17, 18, 19].

Let R be this arithmetic statement of the Riemann hypothesis, and let T be a suitable formal theory in which R has been proved. If R is false, there is a finite computation showing that it is false. Elementary arithmetic can verify that computation and establish $\neg R$. Thus T would contain proofs of both R and $\neg R$. We have

$$\boxed{R \ \vee \ \neg\, \mathrm{Con}(T).}$$

Here $\mathrm{Con}(T)$ means that T is consistent. So if T is consistent, the Riemann hypothesis is true. This follows from the standard result that consistency suffices for the truth of proved Π^0_1 sentences [20]. Appendix A gives the mathematical details, including the conditions on T and the arithmetic formulation.

A stronger theory may prove that T is consistent. But Gödel's second incompleteness theorem tells us that a consistent, effectively axiomatized theory containing sufficient arithmetic cannot prove its own consistency [21]. We can add a statement asserting its consistency as a new axiom. The enlarged theory then faces the same question about its own consistency. Even the effective union of all the finite extensions cannot prove its own consistency, assuming the extensions are consistent [22, 23]. The attempt to obtain an unconditional consistency guarantee by this means has no end.

This is already understood by the people who build proof checkers. Lean's documentation identifies the soundness of its logic as an assumption that remains after checking a proof. The independent checkers used for the Riemann proof have soundness guarantees relative to set theory [24, 2, 25]. Appendix B describes the checks and their assumptions, drawing on the published reports. We have a verified mathematical derivation; the mathematical foundations on which its truth depends have not been eliminated by verification.

There is another question we can ask about the proof. Can we explain why it works? Alpöge himself identifies understanding and communicating the argument as work that remains to be done [1]. Thurston discussed this distinction in 1994: accepting a proof and understanding the mathematics behind it are different achievements [26]. Understanding why a construction works can help us find other problems for which it may be useful. This is part of the reason that the history of a proof matters. Understanding the new proof may lead to other discoveries, including ones outside the original problem.

6 What have we learned about reality?

Once we have a proof, we can ask what it tells us about the world. The Riemann hypothesis is a useful example because one of its consequences is so easy to state. Let $\pi(x)$ be the number of

primes no greater than x and $\mathrm{Li}(x)$ an approximation given by the logarithmic integral. In 1976, Schoenfeld proved that if the Riemann hypothesis is true, then [27]

$$|\pi(x) - \mathrm{Li}(x)| < \frac{\sqrt{x}\log x}{8\pi}, \qquad x \geq 2657. \tag{5}$$

Proving the hypothesis removes the condition from Schoenfeld's result. Not a single prime has moved, and the actual error of the approximation is the same at every x as before. What has changed is what we can guarantee about that error for all x in the specified range.

Korzybski's observation that the map is not the territory helps us see something about this result [28]. The approximation is a mathematical map, but so is the exact prime count. Neither is a measurement of a physical phenomenon. We have learned something about the relationship between two mathematical quantities.

Compare this with using a discrete Fourier transform to study a physical signal. We can prove a result about the transform of an array of numbers without knowing whether the numbers accurately represent the signal. To see the difference, let y be a physical quantity, m the value given by a mathematical model of it, and $\widehat{m}$ a computed approximation to m, all in the same units. Then

$$|y - \widehat{m}| \leq \underbrace{|y - m|}_{\text{model–territory error}} + \underbrace{|m - \widehat{m}|}_{\text{within-model error}}.$$

Computing the model more accurately reduces the second error. It does not establish that the model accurately describes the phenomenon.

The Riemann proof allows us to establish mathematical results that previously depended on the conjecture. If a mathematician had found the same proof without AI, those mathematical consequences would be unchanged. Which of them helps us understand a physical phenomenon or build something better? The use of prime numbers in cryptography, for example, does not establish that a cryptographic system has been improved by the proof. The reports discussed here show what AI-assisted mathematical research can accomplish. Understanding the proof may also reveal techniques useful elsewhere; identifying a practical advance from the theorem itself would be a further achievement.

6.1 Can we tell which mathematics will be useful?

Riemann himself provides an example of why this question is difficult. In 1854, he gave a lecture on geometry that would help make Einstein's theory of general relativity possible more than sixty years later [29]. The mathematics supplied a way to describe curved spacetime. The recognition that it could be used to describe gravitation was a discovery about the relationship between that mathematics and physical reality.

A similar story can be told about Boolean algebra. Boole published his work in 1854. Decades later, Shannon recognized that the algebra could be used to analyze and design electrical switching circuits [30, 31]. Fourier's work on heat conduction shows a different route: mathematics developed to solve a physical problem proved useful far beyond that problem, including in digital signal processing, where the efficient computation of Fourier transforms made a further difference [32, 33]. Work on number theory eventually supplied mathematics used in public-key cryptography, as in RSA [34].

Some mathematics is developed to answer a question about the world. Some begins with a puzzle mathematicians have created, and a physical application is discovered only much later. For other mathematics we still know of no such application. We may not be able to tell, even after centuries, which of these latter two cases we are looking at. Wigner and Hamming explored why mathematics has proved so remarkably useful in science and the role of human choice in that history [35, 36].

We can admire a beautiful result without knowing whether it will ever have a practical use. History gives us good reason to investigate mathematical questions out of curiosity, but it does not tell us that a particular result has already made a difference outside mathematics. Nor does the difficulty of the problem tell us that it will.

7 Which problems matter?

Mathematicians do more than solve problems. They create mathematical objects, decide which relationships to investigate, and discover connections between those objects and the world.

Section 2 established that the supply of difficult mathematical problems is unlimited, even with the proof system fixed. Our time and computational resources are finite. We must decide which of the many problems we can formulate are worth investigating. Knowing how difficult a problem is, even if we can measure it precisely, does not tell us how to make that choice.

Poincaré recognized the importance of selection in *Science and Method*: the mathematician must decide what to study from among the objects mathematicians themselves create [37]. Hamming asked researchers to think seriously about the importance of the problems they had chosen to work on [38]. The ability to solve more problems with AI makes these old questions more pressing.

Recall what Shannon accomplished in Section 6.1. The algebra was already there. He saw that it described the behavior of switches. That observation helped change how we design physical computing systems. Finding a way to describe an important phenomenon can be more useful than proving a very difficult theorem. It is not necessary for the mathematics used to describe the phenomenon to be difficult.

We also pursue mathematics because we enjoy it, as Hardy described in *A Mathematician's Apology* [39]. And because we cannot predict all of its future uses, there is good reason to explore mathematical questions that do not come from a problem in the world. The question is how best to use the resources we have, both to explore and to work on problems that we already know matter.

7.1 What do we want to accomplish in the world?

Finding an application for a mathematical result is not the end of the question. A signal-processing algorithm could make medical imaging cheaper and more accessible. It could also help build systems used to invade people's privacy. In both cases, the mathematics may work very well. To evaluate what we have accomplished, we have to ask whose interests the application serves and what happens when it is put to use.

The reason we ask a question about reality is often that there is something we want to understand or change. We develop mathematical abstractions, test how well they describe reality, and sometimes use them to change it. What we learn from that process can change both the questions we ask and our understanding of what matters to us.

8 Eleven Cognitive Blind Spots

The Riemann case invites us to apply knowledge that we already have. Gödel's results, the role of prior mathematics, the map–territory distinction and the history of mathematical applications all help us assess what the achievement means. The table brings together eleven ways in which we can overlook that knowledge. It identifies possible errors in interpretation, not claims that every error has been made in the public discussion.

	Cognitive blind spot	**What we already know**	**Section**
1	Confusing a checked proof with truth independent of its axioms.	A proof establishes what follows from the axioms. Consistency suffices for the truth of proved Π^0_1 sentences [21, 20, 24].	A.3, B
2	Assuming stronger theories remove the need for assumptions.	A stronger theory may establish the consistency of a weaker one, but cannot establish its own consistency under Gödel's hypotheses [22, 23, 20].	A.4
3	Equating proving a theorem with understanding it.	We can accept a proof without understanding why its argument works [26, 40, 41].	5
4	Mistaking the map for the territory.	A mathematical result does not by itself show that a mathematical model describes physical reality [28, 35, 36].	6
5	Equating difficulty with importance.	A short procedure can generate solvable problems beyond a solver's finite proof-length budget; difficulty alone establishes neither general intelligence nor importance [8, 9, 37, 38, 40].	2, 4, 7
6	Assuming beautiful mathematics describes the world.	Beauty can guide mathematical discovery, but a connection with the world must be found and established [39, 35, 36].	6.1, 7
7	Treating a possible future use as a present benefit.	A potential application does not establish that a benefit has been realized [35, 36].	6.1
8	Assuming mathematics without a known use will never be useful.	Boolean algebra and Riemannian geometry found important applications long after their development [30, 31, 29].	6.1
9	Equating technical effectiveness with human benefit.	We have to ask how a capability is used, whom it helps, and whom it may harm.	7.1
10	Reducing mathematics to solving problems.	Mathematicians also create abstractions, choose questions, explain proofs and find applications [26, 37, 38, 40, 41].	7
11	Giving all the credit to the final solver.	Earlier mathematics can make a proof possible. Elapsed time does not establish the remaining algorithmic difficulty or the contribution of the final solver [1, 10, 11, 12].	3, 4

Table 1. Eleven possible mistakes in interpreting a mathematical achievement.

9 What have we learned, and what remains to be investigated?

The Riemann proof is an achievement in mathematics and evidence of what a human and an AI can accomplish together. Our inquiry has identified several distinct questions about its significance. First, a finite proof-production budget cannot suffice for every solvable mathematical problem in a fixed formal system: we can construct problems exceeding that budget much more easily than we can produce their proofs. Second, the mathematical knowledge accumulated over 167 years may have changed the difficulty faced by the final solver. Third, the proof gives us a new mathematical result within the foundations used to establish it. Its consequences for physical reality and human purposes are separate matters.

The proposed study of difficulty gives us a way to pursue the second question. We can start with the released proof, trace its dependencies, and compare the information and computation needed to produce checked proofs under different libraries. We can then use generated problems and other controlled tasks to investigate what AI can discover with different bodies of knowledge and finite resources. The mathematical construction establishes a limit; the experimental work would help us understand capability within that limit.

If AI reduces the cost of proving theorems, we can investigate many more mathematical questions than before. We can create an unlimited supply of increasingly demanding puzzles, but we can also use that capability to find applications for the mathematics we already have, develop better descriptions of phenomena we do not yet understand, or solve problems whose importance we know from experience. Pure mathematical inquiry remains valuable, both in its own right and because of connections we cannot anticipate.

Thurston's discussion of mathematical understanding, Tao's account of *Math 2.0*, and Martin's discussion of the work of mathematicians all remind us how much of mathematical inquiry consists of activities other than finding proofs [26, 40, 42, 41]. We have the opportunity to devote more effort to understanding the answers we obtain, discovering which mathematics is useful and deciding what questions to ask. The eleven blind spots suggest that applying what we already know is part of that work.

The most significant mathematical question is not necessarily the hardest one we can formulate. It may be the one that allows us to understand or change something in the world that we care about.

A Mathematical foundations

A.1 An arithmetic statement of the Riemann hypothesis

Davis, Matiyasevich and Robinson gave an equivalent formulation of the Riemann hypothesis using rational arithmetic [17, p. 335]. Define, for positive integers j, n,

$$\eta(j) = \begin{cases} p, & j = p^k,\ p \text{ prime},\ k \geq 1, \\ 1, & \text{otherwise}, \end{cases} \qquad \delta(n) = \prod_{1 \leq m < n} \prod_{1 \leq j \leq m} \eta(j).$$

With the empty product equal to 1, their criterion is

$$\text{RH} \iff \forall n \geq 1 : \quad \left(\sum_{k=1}^{\delta(n)} \frac{1}{k} - \frac{n^2}{2} \right)^2 < 36n^3. \tag{6}$$

Each inequality can be checked by finite exact computation. Yedidia and Aaronson describe a Turing machine implementing this criterion [18, Sec. 5]; Matiyasevich gives a register-machine formulation of RH [19].

Let M_R test the inequalities in order and halt if it finds a violation. RH is equivalent to the assertion that M_R never halts. Using the standard arithmetic encoding of finite computations, let $H_R(t)$ mean that t encodes a halting computation of M_R. Then the statement

$$R := \forall t \, \neg H_R(t) \tag{7}$$

is Π_1^0. If R is false, a finite computation witnesses its falsity.

A.2 Why consistency is sufficient

Let T be a classical, effectively axiomatized extension of Peano arithmetic (PA), or a theory with a suitable interpretation of arithmetic. Write $\Pr_T$ for its standard arithmetized provability predicate and $\text{Con}(T) := \neg \Pr_T(\ulcorner 0 = 1 \urcorner)$. We use the usual proof coding and derivability conditions, with PA as a metatheory. To apply the following result to an analytic proof of RH, T must also establish the equivalence with R, either directly or through a translation of the analytic proof.

The standard formalized Σ_1^0 completeness result [20, Lemma 1] gives

$$\neg R \ \longrightarrow \ \Pr_T(\ulcorner \neg R \urcorner).$$

Since proofs of R and $\neg R$ can be combined to give a proof of $0 = 1$, PA proves

$$\Pr_T(\ulcorner R \urcorner) \wedge \neg R \ \longrightarrow \ \neg \, \text{Con}(T). \tag{8}$$

A genuine finite derivation of R in T therefore entails, in standard arithmetic,

$$\boxed{R \ \vee \ \neg \, \text{Con}(T).} \tag{9}$$

Consequently, consistency of T suffices for the truth of R.

A.3 What a theory can establish about its own proofs

Gödel's second incompleteness theorem states that a consistent effective theory T satisfying the conditions above cannot prove its own consistency [21, Theorem XI]:

$$T \nvdash \text{Con}(T).$$

Pudlák gives the standard equivalence between consistency and uniform Π^0_1 reflection [20, Lemma 1]. Here uniform reflection, denoted $\mathrm{RFN}_{\Pi^0_1}(T)$, asserts that every Π^0_1 sentence provable in T is true. The equivalence is provable in an appropriate arithmetic metatheory:

$$\mathrm{RFN}_{\Pi^0_1}(T) \quad \longleftrightarrow \quad \mathrm{Con}(T).$$

Consistency gives reflection by applying the argument of (8) uniformly to Π^0_1 sentences. In the other direction, reflection applied to the false bounded sentence $0 = 1$ gives consistency. Thus a consistent T cannot establish this uniform guarantee internally. The stronger requirement of Σ^0_1 reflection needs Σ^0_1 soundness.

For an individual theorem A, local reflection is different: Löb's theorem gives $T \vdash (\mathrm{Pr}_T(\ulcorner A \urcorner) \to A)$ if and only if $T \vdash A$, under the usual derivability conditions [43]. An individual proved sentence thus admits its local reflection instance even though uniform reflection is not provable in T.

A.4 Why stronger theories do not end the regress

Consider the consistency progression studied by Turing and Feferman [22, 23], with $T_0 = T$ in a fixed language and

$$T_{n+1} = T_n + \mathrm{Con}(T_n).$$

Assume that each stage is consistent. Each stage proves the consistency of every earlier stage, but by Gödel's theorem cannot prove its own. Consistency of T_0 alone is insufficient to ensure this assumption: if PA is consistent, $V = \mathrm{PA} + \neg\,\mathrm{Con}(\mathrm{PA})$ is consistent, yet proves $\neg\,\mathrm{Con}(V)$, so $V + \mathrm{Con}(V)$ is inconsistent. Starting with an arithmetically sound theory preserves soundness at every finite stage [20, Sec. 3].

Taking the union does not solve the problem. Define

$$T_\omega = \bigcup_{n<\omega} T_n.$$

The union is effectively axiomatized because the successive axiom enumerations can be generated uniformly. It is consistent if all the stages are consistent: any finite proof of a contradiction in the union would already be a proof in some finite stage. Therefore

$$\begin{aligned} &T_\omega \vdash \mathrm{Con}(T_n) \quad \text{for each fixed } n, \\ &T_\omega \nvdash \mathrm{Con}(T_\omega). \end{aligned}$$

For the standard uniform encoding, let $C(n)$ express consistency of T_n. PA proves $\mathrm{Con}(T_\omega) \leftrightarrow \forall n\, C(n)$ because each finite proof in the union can be transferred to a finite stage and each stage's proofs are proofs in the union. Thus T_ω cannot prove the universal statement $\forall n\, C(n)$, despite proving every fixed instance.

The same restriction applies at any later stage that is consistent, effective, and contains sufficient arithmetic [44]. All stages retain the original proof of R, so the argument in (8) applies at each stage. A consistency proof may also concern a weaker fragment F that suffices to derive R; the minimal strength required by the actual Riemann proof has not been determined here.

A.5 Generating problems with arbitrarily long proofs

Following Parikh [8, p. 496] and Yanofsky [9, p. 380], fix a consistent extension T of PA with a decidable proof checker and a finite-alphabet proof encoding. Let $P_T(x, n)$ mean that the sentence coded by x has a proof in T of at most n symbols. Checking all strings of length at most n decides

this relation. Use an arithmetic representation such that PA proves each true numerical instance and the negation of each false one.

Use compact numerals $\overline{n}$: read the binary digits of n and successively form terms $2t + b$, with $b \in \{0, 1\}$. Each step adds a fixed number of symbols, so the term has length $O(\log(n + 1))$. Let $s(e, n)$ be the syntactic operation returning the code of the sentence obtained by substituting this numeral into the one-variable formula coded by e. This operation is primitive recursive. The diagonal lemma with a free parameter gives a fixed formula $G(v)$ such that, writing $G_n = G(\overline{n})$,

$$\mathrm{PA} \vdash G_n \iff \neg P_T(\ulcorner G_n \urcorner, \overline{n}).$$

Indeed, apply the parameterized lemma to $\neg P_T(s(e, v), v)$, representing the substitution function arithmetically. The resulting formula asserts the absence of a short proof of its instance at v.

A proof of G_n of length at most n would give $T \vdash \neg P_T(\ulcorner G_n \urcorner, \overline{n})$ by the equivalence. PA would also prove the true numerical instance $P_T(\ulcorner G_n \urcorner, \overline{n})$, contradicting consistency of T. Thus there is no short proof. The negative numerical instance is provable in PA by finite checking, so the equivalence gives $T \vdash G_n$.

Since $G(v)$ is fixed and has only a fixed number of occurrences of v, the generator need only substitute the compact numeral. For every $n \geq 1$,

$$\boxed{|G_n| = O(\log(n + 1)), \quad T \vdash G_n, \quad L_T(G_n) > n.}$$

Here $|G_n|$ is its length in symbols and L_T is the shortest proof length in the fixed encoding. The problem can be written in time linear in the bit length of n after the fixed formula is constructed. If a solver can output at most N proof symbols in this encoding, it cannot solve the task of producing a T-proof of G_N within that budget. This result concerns explicit proofs in the specified calculus; changing the calculus or accepting compressed certificates changes the proof-length measure.

B What Lean verifies

The independent checkers accepted the exported proof, and the statement comparison matched its conclusion to Mathlib's definition of the Riemann hypothesis. Sutherland's audit identified three explicit axiom dependencies [2, 3]:

`propext`, `Classical.choice`, `Quot.sound`.

These are propositional extensionality, classical choice, and quotient soundness. Lean's dependent type theory and its rules also form part of the foundations of the proof.

The Lean Language Reference identifies the soundness of its logic as a remaining assumption after independent verification and distinguishes checking axiom dependencies from establishing consistency [24, 45]. The joint verification statement reports soundness results for independent checkers relative to set theory [2].

Sutherland records acceptance of the proof by ConLeche in verified mode at revision `cd2eceab` [3]. The documentation for that revision specifies a set-theoretic interface with an ω-indexed chain of Grothendieck universes. Conditional on a model of the interface, the checker guarantees a model of the accepted declarations. Its technical overview explicitly discusses this dependence in relation to Gödel's theorem [46, 47, 25]. These assumptions are part of the documented guarantee.

On 24 November, Sutherland also reported checks of Robin's and Lagarias's inequalities, using the Riemann derivation and independently formalized equivalences (project `Robin1984`, revision `5ce18bf3`) [48, 49, 3]. These checks support the connection between the analytic theorem and arithmetic consequences. They use the same foundations, and neither inequality is the exact-rational formulation in (6).

To apply the result in Appendix A, the connection between the checked Lean proof and R must be established in the chosen arithmetized theory T. Given that connection and a valid derivation, consistency of T implies the truth of R. This paper uses the published verification reports; it does not independently replay the Lean development or certify its translation into T.

C Measuring difficulty and testing independent discovery

C.1 Definitions and bounds

Use a fixed effective proof checker for $T + H$, a finite explicitly encoded library H of theorems of T, and a fixed binary encoding of proof certificates. A library citation counts as part of a certificate; its derivation is not charged again. The permitted library must be included in the checker used to construct G_n. Take a machine with a bit-output tape, writing at most one bit per step. Define $t(p, P, H) \geq 1$ as the number of steps to halt with output $\pi = U(p, P, H)$. In (3), $|p|$ is in bits and the minimum is infinite if no proof exists. The definition adapts Krajíček's function [16, Def. 3.1] to a decidable proof relation with an explicit library, without assuming his polynomial-time proof-system results.

For each such program, $t(p, P, H) \geq |\pi| \geq L_T(P \mid H)$, where proof length in this appendix is in bits. Hence

$$\lceil \log_2 L_T(P \mid H) \rceil \leq I_T(P \mid H).$$

A program with a hard-coded witness proof of length m gives the upper bound $m + O(\log(m + 1))$ under an ordinary efficient program encoding. Any actual program producing a proof gives an upper bound by its length and execution time. When its checker is decidable, the existence of a program with score at most a finite integer b is decidable: simulate each program of length $a \leq b$ for at most 2^{b-a} steps, then check any resulting proofs. Thus an exact minimum is in principle computable on provable statements by increasing b until a proof is found. This may be computationally infeasible. The analogous minimization with ordinary unbounded Kolmogorov complexity does not admit this finite test. Our measure depends on the machine, representation and library access protocol; it is not an absolute measure of mathematical difficulty. Although minimum proof length is monotone under nested libraries, information-efficiency comparisons must account for the cost of accessing or simulating a smaller library in a larger one.

Applying the generator directly to binary certificates gives $L_T(G_n \mid H) > n$ and hence $I_T(G_n \mid H) > \log_2 n$. Generating G_{2^b} takes $O(b)$ time once the fixed formula is prepared for that checker and library. The bound holds even for a program that knows the proof construction; a different checker or compressed certificate defines a different task.

C.2 A dependency analysis of the Riemann proof

Fix a version of the formal development, its checker, axioms, inference rules, and common modern encoding. Construct nested libraries H_i of results dated before 1859 or before the announced proof, along with a fixed base library. Exclude RH and results admitted on its assumption. Record historical evidence for each included result and definition. Keep the language fixed rather than trying to simulate historical notation.

Extract the actual dependency graph of the checked proof. For each library condition, recursively replace unavailable references by verified derivations, using the same sharing and counting conventions throughout. Recheck each resulting certificate. For every successful expansion record its length m_i, the code and running time of a reproducible program that generates it, and the resulting upper bounds on $L_T(\mathrm{RH} \mid H_i)$ and $I_T(\mathrm{RH} \mid H_i)$. Record unavailable proofs and uncertain dates. Comparing the expansions quantifies reuse in those particular derivations; it does not establish the optimal reduction or the cost of original discovery. In particular a program written with knowledge of the 2026 proof may contain information that was unavailable in 1859. Its cost is a bound on producing the proof, not an observed historical discovery cost.

C.3 A controlled discovery experiment

For a solver S, define $q_S(P \mid H, r)$ as the probability of producing a checked proof within a resource budget r under a fixed protocol. For a specified success probability $0 < q < 1$, define

$$C_S(P \mid H; q) = \inf\{r : q_S(P \mid H, r) \geq q\}.$$

Use an agreed budget such as total computational work on fixed hardware; also record elapsed time, memory, proof length, verification cost and human assistance. Include all external tools. Estimate probabilities from repeated trials with uncertainty intervals. Report failures at the maximum tested budget, not an infinite measured cost. This empirical measure complements the minimum over programs: it describes what a specified solver can actually discover, without supplying a task-specific program derived from the answer.

Use the generated family for known lower bounds on output length and I_T. For discovery tests, obtain checked witness proofs that fit the budget; record known lower and upper bounds, not exact difficulty labels based on an upper bound alone. Include other tasks with short certificates that require new combinations of ideas. Hold solutions out of the supplied materials, randomize tasks, and compare otherwise identical trials with and without relevant previously proved results. A new instance is not necessarily a new method of solution. Control training exposure when possible, recording the model, training conditions and permitted tools. A library ablation alone measures explicit access to prior results, not independence from human mathematics. A matched range of estimated algorithmic difficulty and observed discovery costs would support comparisons on the tested tasks, not a direct reconstruction of the historical counterfactual for RH.

References

[1] L. Alpöge. Announcement of a claimed proof of the Riemann hypothesis and its Lean formalization (untitled note), 20 November 2026. `https://alpo.ge/remark.pdf`.

[2] Lean FRO and Mathlib Initiative. Lean verification of the Riemann Hypothesis. Joint statement, 21 November 2026. `https://leanprover-community.github.io/blog/posts/riemann-hypothesis/`.

[3] A. Sutherland. Verification files for the Lean proof of the Riemann Hypothesis. Public audit, updated 24 November 2026, 05:25 UTC, including "Arithmetic consequences." `https://math.mit.edu/~drew/grh/`.

[4] Clay Mathematics Institute. Riemann Hypothesis. Millennium Problems, status as accessed 25 November 2026. `https://www.claymath.org/millennium/Riemann-Hypothesis/`.

[5] Clay Mathematics Institute. Rules for the Millennium Prize Problems, revised 26 September 2018. `https://www.claymath.org/millennium-problems/rules/`.

[6] D. Castelvecchi. AI is 'the end of mathematics as we know it'—where does the field go next? *Nature*, 24 November 2026. doi:10.1038/d41586-026-03719-3.

[7] Nature. 'If we can solve Riemann, what can't we do?': mathematicians shocked and excited by AI breakthroughs. 25 November 2026. doi:10.1038/d41586-026-03835-0.

[8] R. Parikh. Existence and feasibility in arithmetic. *Journal of Symbolic Logic* 36(3), 494–508 (1971), p. 496. doi:10.2307/2269958.

[9] N. S. Yanofsky. A universal approach to self-referential paradoxes, incompleteness and fixed points. *Bulletin of Symbolic Logic* 9(3), 362–386 (2003), p. 380. doi:10.2178/bsl/1058448677.

[10] C. Kaliszyk and J. Urban. Learning-assisted automated reasoning with Flyspeck. *Journal of Automated Reasoning* 53, 173–213 (2014). `doi:10.1007/s10817-014-9303-3`.

[11] C. Kaliszyk and J. Urban. Learning-assisted theorem proving with millions of lemmas. *Journal of Symbolic Computation* 69, 109–128 (2015). `doi:10.1016/j.jsc.2014.09.032`.

[12] Claude Fable 5.2, Codex Astra 6.2, and Gemini Carbon, at the behest of M. O. Rubinstein (as credited on the draft). *The Alpöge–Claude proof of the Riemann hypothesis explained for the Riemann zeta function.* Version 2, working draft, 22 November 2026, Secs. I.12 and III.13. Working draft (PDF); Rubinstein's project page.

[13] S. A. Cook and R. A. Reckhow. The relative efficiency of propositional proof systems. *Journal of Symbolic Logic* 44(1), 36–50 (1979). doi:10.2307/2273702.

[14] Riemann-REV project. The condensed proof (25 November 2026): source counts for the original and shortened Lean developments, revisions `40e56bab` and `72fead10`. Public repository, 26 November 2026. `https://github.com/nasqret/riemann-rev`.

[15] A. Sutherland. Verification files for the shorter Lean proof of RH and GRH. Independent verification of Sernau's editions, revision `72fead10`, 26 November 2026. `https://math.mit.edu/~drew/shortgrh/`.

[16] J. Krajíček. Information in propositional proofs and algorithmic proof search. *Journal of Symbolic Logic* 87(2), 852–869 (2022; online 2021), Sec. 3. doi:10.1017/jsl.2021.75. Preprint: arXiv:2104.04711.

[17] M. Davis, Y. Matiyasevich, and J. Robinson. Hilbert's tenth problem: Diophantine equations: positive aspects of a negative solution. In F. E. Browder (ed.), *Mathematical Developments Arising from Hilbert Problems*, Proc. Symp. Pure Math. 28, Part 2, 323–378, AMS (1976).

[18] A. Yedidia and S. Aaronson. A relatively small Turing machine whose behavior is independent of set theory. *Complex Systems* 25(4) (2016). doi:10.25088/ComplexSystems.25.4.297. Preprint: arXiv:1605.04343.

[19] Yu. Matiyasevich. The Riemann Hypothesis in computer science. *Theoretical Computer Science* 807, 257–265 (2020). doi:10.1016/j.tcs.2019.07.028.

[20] P. Pudlák. A note on applicability of the incompleteness theorem to human mind. *Annals of Pure and Applied Logic* 96(1–3), 335–342 (1999), Lemma 1 and Sec. 3. doi:10.1016/S0168-0072(98)00044-X. Author manuscript: `https://users.math.cas.cz/~pudlak/gthm.pdf`.

[21] K. Gödel. *Über formal unentscheidbare Sätze der Principia Mathematica und verwandter Systeme I.* Monatshefte für Mathematik und Physik 38, 173–198 (1931). doi:10.1007/BF01700692.

[22] A. M. Turing. Systems of logic based on ordinals. *Proceedings of the London Mathematical Society* (2) 45, 161–228 (1939). doi:10.1112/plms/s2-45.1.161.

[23] S. Feferman. Transfinite recursive progressions of axiomatic theories. *Journal of Symbolic Logic* 27(3), 259–316 (1962). doi:10.2307/2964649.

[24] Lean Language Reference, version 4.36.0. Validating a Lean proof, especially "Gold Standard: comparator and external checkers" and "Remaining Issues." `https://lean-lang.org/doc/reference/4.36.0/ValidatingProofs/` (accessed 25 November 2026).

[25] ConLeche project. Technical overview (AI-assisted, with the maintainer), Sec. 7, "The set-theory assumption." Revision `cd2eceab`. Version-pinned OVERVIEW.md (accessed 25 November 2026).

[26] W. P. Thurston. On proof and progress in mathematics. *Bulletin of the American Mathematical Society* (N.S.) 30(2), 161–177 (1994). doi:10.1090/S0273-0979-1994-00502-6.

[27] L. Schoenfeld. Sharper bounds for the Chebyshev functions $\theta(x)$ and $\psi(x)$. II. *Mathematics of Computation* 30(134), 337–360 (1976), Corollary 1. `doi:10.1090/S0025-5718-1976-0457374-X`.

[28] A. Korzybski. *Science and Sanity: An Introduction to Non-Aristotelian Systems and General Semantics*. International Non-Aristotelian Library Publishing Co. (1933), p. 58 and Supplement III (1931 lecture).

[29] American Physical Society. 1854: Riemann's classic lecture on curved space. *APS News*, June 2013. `https://www.aps.org/apsnews/2013/06/riemanns-curved-space`.

[30] G. Boole. *An Investigation of the Laws of Thought, on Which Are Founded the Mathematical Theories of Logic and Probabilities*. Walton and Maberly, London (1854).

[31] C. E. Shannon. A symbolic analysis of relay and switching circuits. *Electrical Engineering* 57, 713–723 (1938). doi:10.1109/EE.1938.6431064.

[32] J. Fourier. *Théorie analytique de la chaleur*. Firmin Didot, Paris (1822).

[33] J. W. Cooley and J. W. Tukey. An algorithm for the machine calculation of complex Fourier series. *Mathematics of Computation* 19, 297–301 (1965). doi:10.1090/S0025-5718-1965-0178586-1.

[34] R. L. Rivest, A. Shamir, and L. Adleman. A method for obtaining digital signatures and public-key cryptosystems. *Communications of the ACM* 21, 120–126 (1978). doi:10.1145/359340.359342.

[35] E. P. Wigner. The unreasonable effectiveness of mathematics in the natural sciences. *Communications on Pure and Applied Mathematics* 13, 1–14 (1960). doi:10.1002/cpa.3160130102.

[36] R. W. Hamming. The unreasonable effectiveness of mathematics. *The American Mathematical Monthly* 87(2), 81–90 (1980). `doi:10.1080/00029890.1980.11994966`. Full text: `https://ned.ipac.caltech.edu/level5/March02/Hamming/Hamming.html`.

[37] H. Poincaré. *Science and Method*. Translated by F. Maitland, Thomas Nelson and Sons (1914), Book I, Chapter I, "The Selection of Facts." Originally published as *Science et méthode* (1908).

[38] R. W. Hamming. You and Your Research. Bell Communications Research Colloquium, 7 March 1986; transcript by J. F. Kaiser. `https://www.cs.virginia.edu/~robins/YouAndYourResearch`.

[39] G. H. Hardy. *A Mathematician's Apology*. Cambridge University Press (1940), especially Sections 19 and 25.

[40] T. Tao. Math 2.0. Public lecture, Caltech, 9 October 2026; slides and summary posted 10 October 2026. `https://terrytao.wordpress.com/2026/10/10/math-2-0/`.

[41] U. Martin. 'Mathematics is more than a mechanical process'—why I think the field will survive OpenAI's disruption. *Nature*, 6 November 2026. doi:10.1038/d41586-026-03622-x.

[42] T. Tao. Can maths survive AI? Fields medallist Terence Tao calls for a 'Maths 2.0' world. *Nature*, 22 October 2026. doi:10.1038/d41586-026-03389-1.

[43] M. H. Löb. Solution of a problem of Leon Henkin. *Journal of Symbolic Logic* 20(2), 115–118 (1955). doi:10.2307/2266895.

[44] L. D. Beklemishev. Proof theoretic analysis by iterated reflection. In G. Sommaruga and T. Strahm (eds.), *Turing's Revolution: The Impact of His Ideas about Computability*, pp. 225–270, Birkhäuser, Cham (2015). `doi:10.1007/978-3-319-22156-4_9`.

[45] Lean Language Reference, version 4.36.0. Axioms, Secs. 8.2, 8.4 and 8.5. `https://lean-lang.org/doc/reference/4.36.0/Axioms/` (accessed 25 November 2026).

[46] Lean Kernel Arena. ConLeche checker description and consistency guarantee. `https://arena.lean-lang.org/checker/con-leche/` (accessed 25 November 2026).

[47] J. Breitner and contributors. ConLeche: an external Lean checker with a proof of consistency. Project README and theorem statement, revision `cd2eceab`. Version-pinned README.md (accessed 25 November 2026).

[48] G. Robin. Grandes valeurs de la fonction somme des diviseurs et hypothèse de Riemann. *Journal de Mathématiques Pures et Appliquées* 63, 187–213 (1984).

[49] J. C. Lagarias. An elementary problem equivalent to the Riemann hypothesis. *American Mathematical Monthly* 109(6), 534–543 (2002). doi:10.2307/2695443.