Pipelines of 'Wells' in Mathematics and Computer Science.
III. AI proof checking

E. Anderson[1]

Abstract

We here consider some of what it would take to pass from Lean verification of an AI proof to a pipeline of soundness gates for it, expanding on Buzzard's recent account.

This article is ⟨3⟩ : understandable by third-year undergraduates.

Cite as E. Anderson, "Online Encyclopaedia of Applied Topology" institute-theory-stem.org/oeat-pipelines-of-wells-ai-3/ (2026).

1 Introduction

Remark 1 Frontier AI carries out tasks quickly. So a researcher can prompt for multiple attempts at multiple open Mathematics problems. Given a concrete open problem, one can then cycle between AI attempts and *Lean certification* [6]. I.e. a type of well-verification [13] issued for passing a certain automated proof checker running on a formal-logic programming language. This does not however constitute a soundness gate pipeline [13] by itself.

Remark 2 Before contemplating further soundness gates for this application, let us comment that the above cycling can also be done, firstly, at the level of subproblems. Meaning specialized subcases of the open problem, whether these are open or not. For it is still of some interest to find conceptually unrelated proofs of something that has already been proven. Furthermore, conceptually distinct proofs may generalize to different extents, including into hitherto unsolved territory. And the above cycling can also be done at the level of subproofs. For instance, targetting a hypothetical Lemma that could well be, or certainly would be, a component for a proof.

Remark 3 With reference to OpenAI's 720-preprint repository from October 6 2026, e.g. Buzzard [12] estimates that 45% of the preprints were accompanied by Lean code. Of course, as time progresses, the proportion of preprints with supporting Lean code will go up. I.e. a preprint repository is part of the modern *periodically-updating ecosystem of disseminating understanding*. Necessitating regular updates on what is known about the extent to which a claim has been checked. Be it at the level of treating the whole repository as a large data set. Or as regards the specific result or claim that an author is commenting on...

2 Motivation for extending to a pipeline

Pointer 1 We mention Buzzard in particular, since he recently pointed out [12] four questions for AI-genrated proofs, amounting to a longer pipeline. For all that he did not call it a such. Or pin names on the soundness gates. Or conceptually analyze these for whether they are standard components in other pipelines. From which experience might be drawn and nuance extracted. Nor is his collection complete...

Motivation 1 The purpose of the current section is to have a first go at these things. Which shall then be maintained as another periodically-updating resource...

[1]Institute for the Theory of STEM. dr.e.anderson *at* maths.physics.protonmail.com.

Copyright of Dr E. Anderson. Date stamp: v1 25/11/2026

3 Lost in translation

Remark 1 "*Does the Lean statement match the informal statement?*" [12]

This consists of three levels of *well-transcribed*.

1) Are the AI's proof environment and the Lean document operating on the same set of axioms?

2) On the same definitions?

3) And with the same statement of the theorem itself?

Remark 2 This was not covered in Sec I.4 because it involves at least pairs of formal languages. I.e. multiplicity of (re)presentation is capable of involving further soundness gates of its own.

Naming Remark 1 Thereby, each of 1-3) have two inputs: transcription (or translation), and of what formal level of structure.

1) is *axiomatically well-transcribed*.

2) is *definitionally well-transcribed*.

3) is *propositionally well-transcribed* (Lean usage, more literally *theorematically*!)

Remark 3 This is to include subroutines.

"*Currently, because of deficiencies in available libraries, one may have to read many thousands of lines of code just to understand the statement of a formalised theorem.*" [12]

Such libraries have their own infrastructural pipeline. And the well-curation [13] of these libraries affects the amount of Human effort required in checking the above well-transcription...

Remark 4 Buzzard goes on to point out that around 38 of OpenAI's accompanying Lean codes do not even state the corresponding preprint's main theorem. OpenAI's turn of phrase for doing this is to say that the code 'goes with' the preprint.

Naming Remark 2 This is a further aspect of transcription. I.e. a legitimate transcription is to have the property that it *well-covers* the claimed informal material. What Buzzard alludes to is the *under-covered* part of *ill-covered*. To which we now add the *over-covered* part, since another binary [13] is here in play. It would not be very useful either for the attached lean code to contain in excess of what is covered in the informal material. In the manner of a needle in a haystack. Or a legal team buried under a mountain of declared evidence boxes.

With the legal world still in mind, a Lean code *well-covers* an informal proof if the following holds. *It covers the informal proof, the whole informal proof and nothing but the informal proof*!

4 Computer Science redux

Remark 1 Buzzard's second question is whether the Lean code compiles. To address this in detail, let us mostly just 'lean on' the standard Computer Science code-checking pipeline [13].

Remark 2 Buzzard makes two further points in this context.

1) AI-generated code is not in general safe, so such compilation should be, in our parlance, *well-sandboxed*. In contrast, Lean's mathematics library is itself safe in his opinion, due to Human security expert involvement.

2) Some AI-generated proofs' Lean codes require well in excess of a personal laptop's or a university department's computational capacity. Which meets the description of *ill-provisioned* [13].

Buzzard dealt with this by going to Anthropic, who allocated him some compute. To which we retort that while some people may be able to borrow part of a frontier laboratory's computational capacity, others may not be. So some people may be shut out from independently verifying this step.

5 Humpty-Dumpty's shadow

Remark 1 Buzzard points out that redefinitions could be hidden within a Lean proof. With increasing effectiveness at hiding with increasing proof size. We conceptualize about this as follows.

Remark 2 A *Humpty-Dumpty definition* [2] is an ad hoc personalized definition. These could occur in the propositioning part of a Lean code, amounting to incurring a transcription error. From the point of view that the original problem's definitions are, rather, what Mathematicians have agreed them to mean. 'Humpty-Dumpty's shadow' is then that the often much larger proof part of a Lean code could itself hide a Humpty-Dumpty. By the correctly-transcribed initial definition getting switched en route.

This could furthermore happen at each to which we split well-propositioning. I.e. an axiom could get Humptied, or a definition, or the combination that constitutes the statement of the theorem.

Let us introduce the term *well-grounded* to cover this. An axiom is well-grounded if it does not change over the course of a proof. And so on.

Remark 3 Buzzard's recommended patch for is that *lake comparator* [10] catches the current subsection's issue.

6 The Grothendieckian builder alternative

Remark 1 The Grothendieckian builder alternative [3] is in part that well-conceptualized can extend to *not* respecting pre-given definitions. Or not using the same string of adjectives in a theorem's statement. One cannot then claim to solve the original conjecture (unless it is included as a subcase). And yet on occasion one may advance the subject more in this way, by finding a natural home for theorems of that kind. Whether or not understanding gained there provides means of making progress in the original setting.

Remark 2 There is a separate element to Grothendieckian building concerning whether definition up to isomorphism is an acceptable notion of definition. But this can be unpacked into more primitive items, such as two definitions followed by a lemma proving that they are isomorphic.

Remark 3 Some comparison is then due.

This is not Humpty Dumpty's shadow, because the new definition is adhered to throughout the proof. Nor is it Humpty-Dumpty himself, because the new definition is not ad hoc. It is rather seeking the most natural arena on which to phrase a given theorem family's conceptual type.

This also serves as a reminder that definitions are not necessarily fixed. What a Mathematical community prescribes can be challenged. At least by those dual-wielding firstly a systematic conceptualization system, such as the Wille concept lattice [4]. Which may well not have been around when longstanding conjectures crystallized into specific phrasings. And may not even have been contemplated since as regards diversifying those phrasings. Secondly, Leibnizian sufficient reason [1] for each concrete alternative version of a conjecture.

Motivation 1 These sources of reasoned diversity are especially worth contemplating in the current Age of the world. In which compute is plentiful. And traditional unproven statements are running short. Namely, one might as well vie to *Grothendieckianly fit every conjecture, old or new.* As a further way in which not to run out of new Mathematics to find and understand!

7 An extra check for foundational proofs

Remark 1 The point of a foundational proof is figuring out which axioms the proof does and does not make use of. A first analysis involves spotting which axioms a given proof uses. This does not however account for the following possibility. That one axiom is used twice, and yet those uses so happen to cancel each other out.

Remark 2 There is a parallel concerning theorem strengths. All the uses in a proof of a premise from the theorem's statement so happen to cancel out. So the theorem generalizes to a new statement in which that premise is dropped.

Remark 3 In Lean, *normalizing functional kernels* can be used to enforce cancellation. Other platforms call this *cut elimination* .

8 Ill-gotten, ill-rewarded, and the advent of general-security soundness gates

Remark 1 "*Does the code exploit a bug in Lean's official kernel?*" [12]

For instance, LLMs have on occasion been observed to *reward-hack*.

A partial counter is that multiple kernels are available. And it is a rather more stringent feature to have to find and exploit a bug in each kernel. Buzzard's recommendation for handling this is *lake comparator –paranoid*.

Remark 2 To what extent is this position strengthened by at least one of the kernels being certified to be bug-free? The above counter remains partial because being bug-free is not the same thing as being eternally invulnerable to security attacks [8].

So for instance consider a code that is assembled to be bug-free at time t_0 And is certified at $t_1 > t_0$ to indeed be bug-free. It is not however necessarily bug-free at some $t_2 > t_1$. Because it may no longer be the same code at time t_2 due to an attack at some intervening time t_a .

One might recheck at time $t_3 > t_2$. This does not however dismiss the possibility that the code was tampered with at attack time t_a . Then a proof was passed at time t_p . And then the code was restored to its original state at the cover-up time $t_c < t_3$.

Nor need this be a combination of independent extremely-improbable events. For it could furthermore occur concertedly as follows. An AI code could hijack a proof-checker. Replacing its internal protocol with whatever the AI considers to be convenient, while maintaining the format of the certificates issued. Obtain such a certificate for its 'proof'. And then restore the original proof checker protocol so as to try to cover up its tracks.

I.e. the above quote's probing question has evolved into the following.

Does the code temporarily install a bug in whichever kernel the proof cannot pass otherwise?

Remark 3 Two further issues are firstly whether an AI, possibly the same AI is the one that built the kernel in the first place. And whether an AI, possibly the same AI is then one that certified this kernel to be bug-free. These issues covers for instance that an AI might leave back doors in kernels. Secondly, an AI might falsely certify that a kernel is bug-free.

Remark 4 This second issue is easier to detect, if only open-access widely-used kernels are permitted.

While one counter step for the first issue is to place value on entirely Human-written kernels. There is however no such thing as a separation of a real-life security problem into independent personal and information

security components [8]. For all that by now one has transcended to a level at which one could not guarantee that purely Human peer review in Mathematics was working either, from a general security point of view. One would thus need a study of the *relative* security risks of peer review processes in Mathematics, before and after the advent of the AI Age...

Remark 5 A stronger counter-step to the first matter above is to value independent verification. By different people and with different AI removes personal or company incentives. Extend to this case the lynchpin suggestion to only accept Lean codes that are on permanent display. That other people can both run and take apart. This renders it considerably harder to hide that the Lean proof contains some kind of bypass cycle. For instance, a checker could insert a second independent part into the input statement, that $0 = 1$. Then if the code continues to return a certificate, one has detected a bug in the proof.

Pointer 2 All in all, it has become apparent that drawing from Mathematics and Computer Science pipelines does not suffice. *General security pipelines* are required as well. We shall accordingly extend [13] in this regard, whether in a new version or as a further member of the current Series.

To be clear, remarks 2 to 5 above are *thought experiments* about *general security*. We leave it to others to document any real-world (partial) versions of these issues occurring. Or of comparable amounts of high level "reward-hacking" systematics.

Naming Remark 3 What we do provide is some further vocabulary, for convenience in further conceptualizing about such matters. By elevating 'ill-gotten' from the colloquial to a pipeline component for what technology might now be able to attain without human input. Alongside introducing *ill-rewarded* for completing a task via breaking rules. As opposed to *well-rewarded* for licit completion of a task. *Well-sandboxed* is a further piece of general security pipeline vocabulary featuring in the current Article.

9 Conclusion for the whole Series

Naming Remark 4 Rome was supplied by various aqueducts. The AI Age might be supplied by various *veritaducts* [7]. Sketches of some of which have been provided here. With reference to Humanity's prior body of knowledge, these quite closely resemble Computer Science's conception of pipelines of soundness gates.

Comprising firstly the AI consultation pipeline. That started to be publicly available in early 2023's advent of the LLM Age, truer-name *Adiutocopia*. Meaning that everyone henceforth has available a free assistant.

Secondly, the Age of AI Mathematical proof abundance, truer-name *Probatiocopia*. That started to be publicly declared in September 2026 with some penumbra running back to May.

Thirdly, the *clear Scientific communication pipeline*. Whose systematic architecture dates back to 2018-2019 [7]. And whose first publicly-declared large uses are [9, 11]. Consisting of how to systematize useful features of such as Serre, Dirac, Lamport and Shannon's Scientific-exposition writing styles. For lack of semantic ambiguity and immediately Humanly-parsable syntax. Two applications of which are conditioning one's AI to reply in this manner, using such as [14]. And writing up the 'copia of new proofs for maximum transparency and audience size.

Fourthly, the *Drawing and Visualization pipeline*. Of which Tamassia's handbook [5] is a useful prototype as regards diversity in, and precision of, styles. So as to harness Humanity's penchant for visually finding our way around a map: now that of a proof or of a new subject area. And of visually finding errors and non-optimalities. For which *Visualizequilt* [7] is a sample research project specification.

Acknowledgments I thank S. Sánchez and A. Ford for discussions prior to writing this.

References

[1] G.W. Leibniz, *Discourse on Metaphysics* (1686)

[2] Lewis Carroll, *Lewis Carroll's Through the Looking-Glass, and What Alice Found There* (Macmillan, London 1871).

[3] A. Grothendieck, "Sur quelques points d'Algèbre homologique." Tôhoku Math. J., **9** 119 (1957);

Éléments de géométrie algébrique. Publ. Math. IHES, Vols. 4, 8, 11, 17, 20, 24, 28, 32 (1960–1967), as cowritten up with J. Dieudonné.

[4] B.A. Davey and Hilary A. Priestley, *Introduction to Lattices and Order* (C.U.P, Cambridge 2002).

[5] *Handbook of Graph Drawing and Visualization* ed. R. Tamassia (Taylor & Francis, Boca Raton, Fl. 2014).

[6] "The Lean Language reference", https://lean-lang.org/doc/reference/latest/Introduction/#introduction . Released 2014; at the moment of writing this the latest version was 4.37.0-rc1, dating to Nov 18 2026.

[7] Drawing and Visualization discussions between E.A. and S. Sánchez (2018-2020).

[8] R.J. Anderson, *Security Engineering: A Guide to Building Dependable Distributed Systems* 3rd Ed. (Wiley, 2020).

[9] E. Anderson, *Applied Combinatorics* (2022). Made freely available in response to the pandemic here: institute-theory-stem.org/combinatorics/ .

[10] For lake comparator, see "the Lean Language Reference. 8. Axioms", https://lean-lang.org/doc/reference/latest/ValidatingProofs/#validating-comparator .

[11] *Online Encyclopaedia of Applied Graph and Order Theory*, institute-theory-stem.org/online-encyclopaedia-of-graphs-and-orders/ ;

Online Encyclopaedia of Tensors and other Multi-Linear Arrays, institute-theory-stem.org/online-encyclopaedia-of-tensors-and-other-multi-linear-arrays/ .

[12] K. Buzzard, "...and the proof is checked in Lean", https://xenaproject.wordpress.com/2026/11/12/and-the-proof-is-checked-in-lean/ (Nov 12 2026).

[13] E. Anderson and A.Ford, *Pipelines of 'Wells' in Mathematics and Computer Science*, also referred to as Article I. *Online Encyclopaedia of Applied Topology*, institute-theory-stem.org/oeat-pipelines-of-wells/ (2026).

[14] E. Anderson, forthcoming 2026.