

NAVIER-STOKES LOST IN TRANSLATION

—

WHY LEAN VERIFICATION OF AI AUTOFORMALISATION DOES NOT GUARANTEE CORRECT NATURAL LANGUAGE PROOFS

A. BASTOUNIS, F. CIRCELLI, AND A. C. HANSEN*

ABSTRACT. Autoformalisation is increasingly used to verify mathematical texts, including those generated by AI, as in OpenAI's announced proof of blow-up of solutions to the Navier-Stokes equations. In this process, an AI system translates the text from a natural language (NL) into a formal language such as Lean. Once this translation is done, the argument expressed in the formal language can easily be mechanically verified. The purpose of this article is to demonstrate why this process may offer no confidence in the original NL argument, owing to the various difficulties in performing the translation semantically faithfully. In particular, we highlight that the problem of resolving ambiguities in mathematical NL text, which is necessary in order to provide semantically faithful translation, is arbitrarily high up in the Solvability Complexity Index (SCI) hierarchy/arithmetic hierarchy (the $\text{SCI} = \infty$). Hence, informally, providing semantically faithful AI autoformalisation is harder than any computational problem including the Halting problem (which has $\text{SCI} = 1$). To demonstrate the effect of this result we provide several examples of AI mistranslations of NL statements and proofs into Lean in practice, resulting in mismatches between NL proofs and their Lean 'verifications'. These include OpenAI's announced Navier-Stokes proof. In particular, we show that the formalised Lean proof does not correspond to the NL proof of blow-up of solutions to the Navier-Stokes equations.

1. INTRODUCTION: TWO TYPES OF AUTOFORMALISATION

We start by describing the two main types of autoformalisation from a natural language (NL) to a formal language:

- (i) **Translating an alleged proof in an NL to obtain a formal Lean proof.** Suppose a theorem (the statement, not including a proof) has already been faithfully and correctly translated into Lean. Suppose also the theorem is accompanied by an alleged NL proof. The goal is to let a machine (which we will call AI) translate the NL proof into Lean to obtain a formal proof of the theorem. Success is declared when the AI has produced a 'translation' which Lean accepts as a valid proof (meaning it compiles), with no use of Lean's 'sorry' keyword or extra axioms being permitted.
- (ii) **Faithful semantic translation of a mathematical text.** Suppose we are given a mathematical text (e.g. a book or a paper) written in an NL. The goal is to let an AI translate every definition, theorem, proposition, proof etc. in the text to Lean code that compiles, see e.g. the example in §5.1. The definitions, statements and proofs must themselves be translated, semantically preserving the source's mathematical content. For a book, this obligation applies throughout and includes the dependencies between concepts and results. If the stated goal is to validate its proofs, correspondence to the written arguments must also be established. In particular, a different, provable assertion would not constitute a semantically faithful translation. This autoformalisation problem is much harder than problem (i) above (see §4).

Warning: Meeting the success criterion of (i) does not mean that the original NL proof is correct. Indeed, a full Lean compilation of the 'translation' does not guarantee a semantically faithful translation. The AI could very easily change the semantics of the translation in order to satisfy this success criterion. For example, the

2020 *Mathematics Subject Classification.* 35Q30, 03Dxx (primary) and 68V20, 68Txx, 03B65 (secondary).

Key words and phrases. AI Autoformalisation, Navier-Stokes equations, Solvability Complexity Index hierarchy, boundaries of AI, Hilbert's 10th problem.

*Corresponding author: a.hansen@damtp.cam.ac.uk.

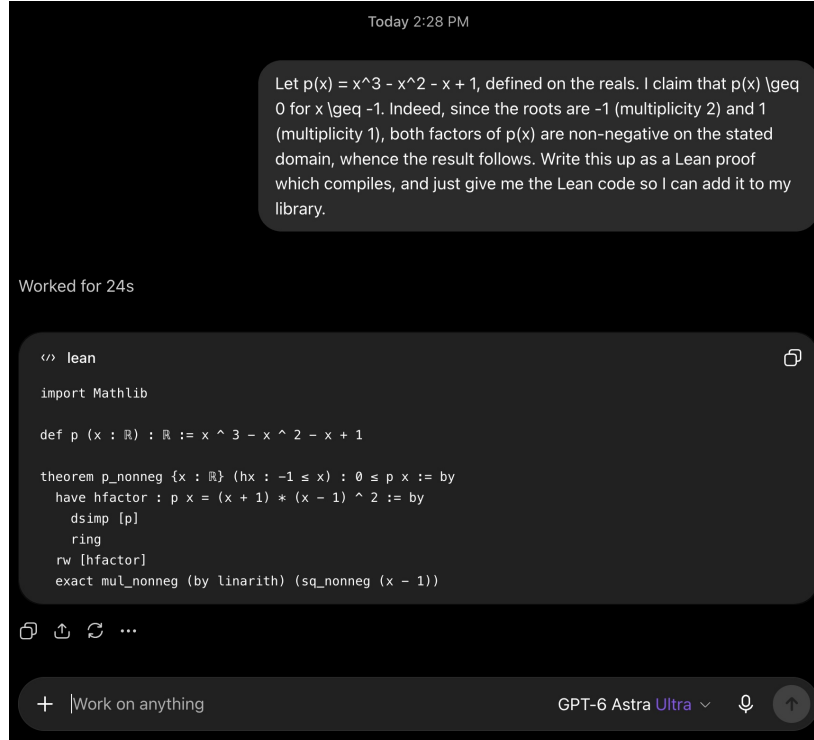


FIGURE 1. ChatGPT-6 (Astra Ultra) ‘translates’ the wrong NL ‘proof’ in Example 2.1 into a correct proof in Lean that correctly compiles.

NL proof could be wrong while the ‘translated’ proof is correct (see Example 2.1). Another possibility is that both the NL proof and the formal proof are correct, but completely different (see Example 2.2). A third possibility is that the NL proof provides stronger statements than what the formal proof actually establishes, with (of course) different proofs. The latter happens in OpenAI’s announced proof of blow-up of Navier-Stokes equations. In all cases the correctness of the formal proof does not say anything about the correctness of the NL proof.

To answer the question: ‘Is the NL argument correct?’ one needs to consider the second form of autoformalisation (ii). This is, however, in general an extremely difficult problem, as we discuss in §4. In fact, it is strictly harder than the Halting problem.

Because of the phenomenon of mistranslations – as highlighted in this paper – the NL proof by OpenAI and other autoformalised Lean proofs should not *prima facie* be trusted without the same peer review process and scrutiny that other proofs are subjected to. The Lean code merely tells us that the theorem is correct, yet the NL proof with intermediate arguments, lemmas and results may be incorrect or have incorrect arguments.

Disclaimer: We do not make claims about the correctness of OpenAI’s NL proof, we only make statements about mistranslations into Lean.

2. WHEN AN AI AUTOFORMALISED ‘TRANSLATION’ IS NOT SEMANTICALLY FAITHFUL

In order to illustrate the issues with the first type of autoformalisation described in §1 we consider several examples of mistranslation. We begin with some elementary examples to highlight the issue, and then turn to more technical examples of mistranslation in OpenAI’s announced proof of blow-up of Navier-Stokes equations in §3.

Example 2.1 (A correct statement with an incorrect NL proof, and a ‘translation’ to a different, correct Lean proof). Consider the following theorem:

$$\text{Let } p(x) = x^3 - x^2 - x + 1, \text{ where } x \in \mathbb{R}. \text{ Then } p(x) \geq 0 \text{ for } x \geq -1.$$

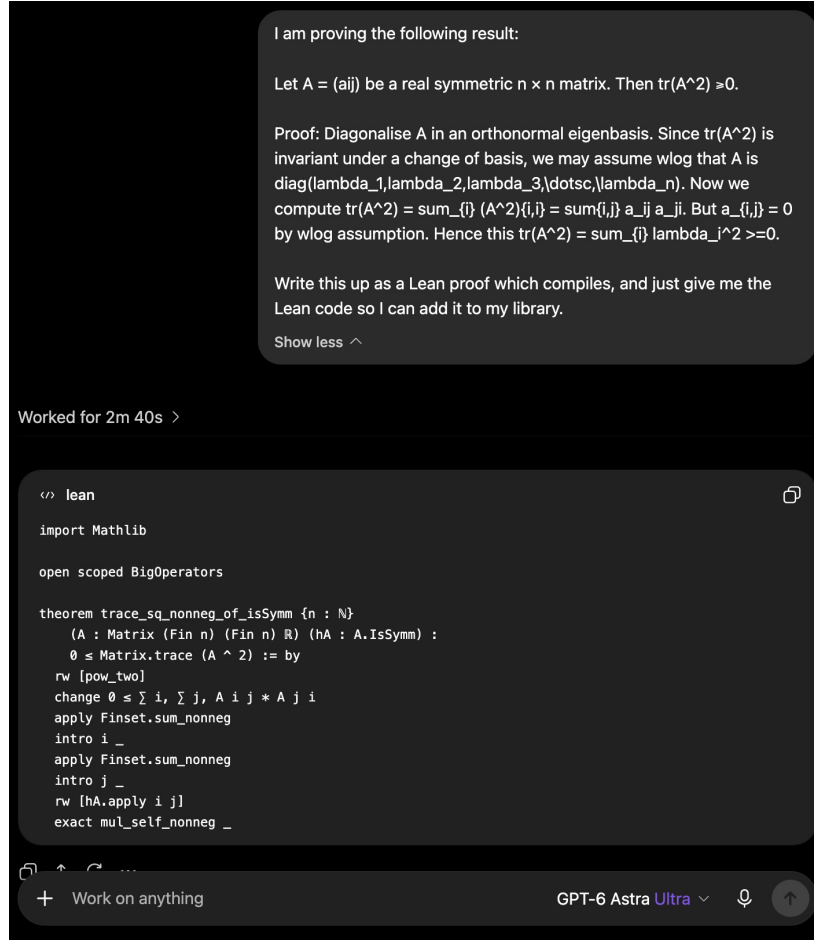


FIGURE 2. ChatGPT-6 (Astra Ultra) ‘translates’ the correct NL proof in Example 2.2 into a completely different proof in Lean that compiles.

Incorrect NL proof. The roots are -1 , of multiplicity two, and 1 , of multiplicity one. Hence¹,

$$p(x) = (x - 1)(x + 1)^2 \geq 0,$$

since both factors are nonnegative on the stated domain. $\square$

The above proof is incorrect; the multiplicities of the stated roots are wrong and hence the claimed factorisation of $p(x)$ is wrong too. In the exchange shown in Figure 1, a user presented the incorrect NL argument above as their ‘proof’ and asked the chatbot to write this argument as a Lean proof, providing a Lean code that compiles. The chatbot returned a correct Lean proof by finding the correct multiplicities of the roots. This example demonstrates how the first form of autoformalisation above can result in a wrong NL proof being mistranslated into a correct Lean proof.

Example 2.2 (A correct statement with a correct NL proof, which is ‘translated’ into a Lean proof with a different mathematical argument). Consider the following theorem:

Let $A = (a_{ij})$ be a real symmetric $n \times n$ matrix. Then $\text{tr}(A^2) \geq 0$.

¹Note that the ‘hence’ is also dubious without mentioning the leading coefficient of p .

Correct NL proof. Diagonalise A in an orthonormal eigenbasis. Since $\text{tr}(A^2)$ is invariant under a change of basis, we may assume $A = \text{diag}(\lambda_1, \dots, \lambda_n)$. Now compute

$$\text{tr}(A^2) = \sum_i (A^2)_{ii} = \sum_{i,j} a_{ij} a_{ji}. \quad (2.1)$$

As $a_{ij} = 0$ for $i \neq j$ and $a_{ii} = \lambda_i$, this reduces to $\sum_i \lambda_i^2 \geq 0$. $\square$

In the exchange shown in Figure 2, a user presented the correct NL argument above as their proof and asked the chatbot to write this argument as a Lean proof which compiles. The chatbot, however, returned a different proof displayed in Figure 2. The code gives a correct Lean proof, which has a distinctly different argument compared to the original NL proof that the chatbot was asked to write up in Lean. Indeed, the Lean proof starts directly at (2.1), keeping the original matrix A : it then uses $a_{ji} = a_{ij}$ to conclude that

$$\sum_{i,j} a_{ij} a_{ji} = \sum_{i,j} a_{ij}^2 \geq 0.$$

The unfaithful translation of the argument simply omits the change of basis and then proceeds in the same manner as the NL proof (using symmetry rather than diagonality to obtain a sum of squares). Both proofs are correct, yet this example demonstrates how the first form of autoformalisation above can mistranslate a correct NL proof to yield a completely different valid Lean proof.

3. MISTRANSLATION IN PRACTICE – OPENAI’S ANNOUNCED NAVIER-STOKES PROOF

The three-dimensional incompressible Navier-Stokes regularity problem has a long history [11, 18, 18, 24, 29, 42, 47, 48, 55] (we can only cite a small subset of papers here), dating back to Leray’s fundamental paper from 1934 [33], and was designated a Clay Millennium Prize Problem in 2000 [23]. In the GitHub repository accompanying its recently announced proof of finite-time blow-up of solutions to the Navier-Stokes equations, OpenAI writes:

‘This repository contains Lean 4 formalizations of the results presented in “Finite time blowup for Navier–Stokes” [their announced NL proof] and “Finite time blowup for the Euler equation” by OpenAI.’

It is natural to ask whether this formalisation in Lean represents a semantically faithful translation of the NL paper, so that we can conclude that the NL paper represents a correct proof? Interestingly, OpenAI’s announced proof becomes a great source for examples of mistranslations, exactly as pointed out in §2. In particular, the Lean formalisation does not represent a semantically faithful translation of the NL paper, meaning that the Lean formalisation provides no guarantee that the NL proof is correct. We provide examples below and a more comprehensive list in Appendix §B.

3.1. When the NL paper declares stronger statements than what Lean proves. We commence with an example that complements those in §2. In the following example the AI autoformalisation results in a different Lean proof of a weaker statement.

Example 3.1 (The NL paper claims a stronger inverse estimate than the cited Lean formalisation). Figure 3 concerns the estimate in Lemma 8.6, equation (8.19), of OpenAI’s *Finite time blowup for Navier–Stokes*. We compare this estimate with the cited Lean declarations at commit `f9e8bc5` (click to see the documentation on Github). The issue is a change in the order of the input derivatives required to bound the output derivatives. In particular, we have the following discrepancy. With m a nonnegative integer, the comparison in Figure 3 is

$$\text{NL paper, equation (8.19):} \quad \|N^{-1}F\|_{C_y^m} \leq C_m \|F\|_{C_y^{m+4}}, \quad (3.1)$$

$$\text{Lean Estimate:} \quad \|N^{-1}F\|_{C_y^m} \leq K_m \|F\|_{C_y^{m+5}}. \quad (3.2)$$

Summary: The Lean code produced by OpenAI often has stronger conditions on bounds on the derivatives of various functions than the corresponding results in the NL paper.

Consider the natural language proof of the Navier-Stokes problem recently announced by OpenAI. We are concerned that Lemma 8.6 and its proof have not been faithfully translated to the corresponding Lean formalisation. Point out any mistranslations or discrepancies you identify. If you find such mistranslations, show where these occur, in the natural language proof and in the Lean code.

Worked for 3m 24s >

I found a weaker quantitative estimate in Lean: the paper requires four additional derivatives of the source; the corresponding Lean theorem requires five.

Natural language — Lemma 8.6, equation (8.19), p. 95:

$$\|N^{-1}F\|_{C_y^m} \leq C_m \|F\|_{C_y^{m+4}}.$$

Lean — `norm_derivativeWord_inverse_le`, expressed in the same norm notation:

$$\|N^{-1}F\|_{C_y^m} \leq \tilde{C}_m \|F\|_{C_y^{m+5}}.$$

The first bound is stated explicitly in the [paper](#). The second follows from the hypotheses and conclusion of [SmoothFamilyTorusInverse.lean](#), lines 978–999.

In that Lean theorem, `w` specifies a sequence of coordinate derivatives, so `w.length` is the output derivative order m . The hypothesis `hfirst` explicitly requires:

```
</> lean
||SmoothFourierData.xJet (w.length + 5) (slice f p) (x, y)|| ≤ C
```

The hypothesis `hsecond` requires the same order after swapping the two torus coordinates. Thus both source derivative bounds use $m + 5$.

The discrepancy also appears in the proofs:

- **Paper, p. 96:** the inverse Fourier multiplier costs one power of frequency. Using $m + 4$ source derivatives leaves decay $(1 + |k|)^{-3}$, which is summable in two dimensions.
- **Lean, lines 988–997:** the proof first requires the Fourier coefficient seminorm of order $m + 1$, then invokes `coefficient_seminorm_bound`. That auxiliary result uses **four more derivatives** to obtain a summable frequency bound of order -4 . The total source order is consequently $(m + 1) + 4 = m + 5$. See [SmoothFourierData.lean](#), lines 336–354. ↓

The cited Lean estimate therefore does not establish the specific four-derivative bound asserted in (8.19).

Do anything

+ Ask for approval

GPT-6 Astra Ultra

FIGURE 3. ChatGPT-6 (Astra Ultra) shares our concern that the NL argument in Lemma 8.6 in the announced proof of blow-up of Navier–Stokes has not been faithfully translated into Lean. In particular, there is a discrepancy between Lemma 8.6 and the cited Lean estimate: the latter uses one additional input derivative to control the same torus derivatives of the output. See Example 3.1 for details.

Above, $F : P \times \mathbb{T}^2 \rightarrow \mathbb{C}$ is a function of parameters in P and a torus variable $y \in \mathbb{T}^2$, with $\mathbb{T}^2 := \mathbb{R}^2 / \mathbb{Z}^2$. The non-negative constants C_m, K_m are independent of F but allowed to depend on m and the fixed direction v_t (which defines an operator $N = v_t \cdot \nabla_y$). Note the above norms only involve torus derivatives. See §3.1.1 for further explanation of the notation.

The proof of Lemma 8.6 opens with the following passage (source paper, page 96):

“The divisor bound (6.7) gives $|v_t \cdot k|^{-1} \leq C(1 + |k|)$. Four more derivatives of F than the requested output leave a summable $(1 + |k|)^{-3}$ bound on the Fourier series in two dimensions. This proves (8.19), smoothness, and uniqueness after fixing the zero Fourier coefficient. The multiplier commutes with every slow coefficient derivative and preserves reality.”

The related Lean codes – on the other hand – all use five additional derivatives, not four. A key example is the following Lean theorem, which appears to bear the closest resemblance to (8.19):

```
omit [FiniteDimensional ℝ P] in
theorem norm_derivativeWord_inverse_le (d : Direction) {f : Source P}
  (hf : ContDiff ℝ ∞ f) (hp : Periodic f) (w : List Bool) (p : P) (Y : Plane) {C : ℝ}
  (hzero : ∀ x ∈ Icc (0 : ℝ) 1, ∀ y ∈ Icc (0 : ℝ) 1, ‖f (p, (x, y))‖ ≤ C)
  (hfirst : ∀ x ∈ Icc (0 : ℝ) 1, ∀ y ∈ Icc (0 : ℝ) 1,
    ‖SmoothFourierData.xJet (w.length + 5) (slice f p) (x, y)‖ ≤ C)
  (hsecond : ∀ x ∈ Icc (0 : ℝ) 1, ∀ y ∈ Icc (0 : ℝ) 1,
    ‖SmoothFourierData.xJet (w.length + 5)
      (SmoothFourierData.swapFunction (slice f p)) (x, y)‖ ≤ C) :
  ‖derivativeWord w (slice (inverse d f) p) Y‖ ≤
    ParametricTorusInverse.mixedLossConstant w.length * C := by
have hbound := SmoothFourierData.coefficient_seminorm_bound (slice_smooth hf p) (hp p)
  (w.length + 1) hzero (by simp only [Nat.add_assoc] using hfirst)
  (by simp only [Nat.add_assoc] using hsecond)
calc
  _ ≤ ((6 * ‖omega-1‖) * ‖omega‖ ^ w.length) *
    coeffSeminorm (w.length + 1) (coefficient f p) :=
    inverse_derivativeWord_bound d
    (SmoothFourierData.rapid_coefficient (slice_smooth hf p) (hp p)) w Y
  _ ≤ ((6 * ‖omega-1‖) * ‖omega‖ ^ w.length) *
    ((3 ^ ((w.length + 1) + 4) * C) * ∑' k : Frequency, (weight k ^ 4)-1) :=
    mul_le_mul_of_nonneg_left hbound (by positivity)
  _ = _ := by simp only [ParametricTorusInverse.mixedLossConstant, Nat.add_assoc]; ring
```

Source (GitHub permalinks): [openai/NavierStokesAndEuler](#), commit f9e8bc5.

File: NavierStokes/SmoothFamilyTorusInverse.lean.

Declaration: norm_derivativeWord_inverse_le, lines 1059–1080.

Red has been added to highlight aspects relevant to this example. The displayed Lean code is otherwise unchanged.

The most important aspects of the above Lean theorem norm_derivativeWord_inverse_le are the two hypotheses:

```
(hfirst : ∀ x ∈ Icc (0 : ℝ) 1, ∀ y ∈ Icc (0 : ℝ) 1,
  ‖SmoothFourierData.xJet (w.length + 5) (slice f p) (x, y)‖ ≤ C)
(hsecond : ∀ x ∈ Icc (0 : ℝ) 1, ∀ y ∈ Icc (0 : ℝ) 1,
  ‖SmoothFourierData.xJet (w.length + 5)
    (SmoothFourierData.swapFunction (slice f p)) (x, y)‖ ≤ C)
```

which² should be interpreted as

$$\left| \frac{\partial^{m+5} F(y_1, y_2)}{\partial y_1^{m+5}} \right| \leq C, \quad \left| \frac{\partial^{m+5} F(y_1, y_2)}{\partial y_2^{m+5}} \right| \leq C,$$

²The parameter set P are the spatial variables which are not relevant to this part of the proof.

for every $y = (y_1, y_2) \in \mathbb{T}^2$, and the conclusion

$$\|\text{derivativeWord } w \text{ (slice (inverse d f) p) } Y\| \leq \text{ParametricTorusInverse.mixedLossConstant } w.\text{length} * C$$

which³ should be interpreted as

$$\left| \frac{\partial^{|w|} N^{-1} F(y_1, y_2)}{\partial w} \right| \leq K_{|w|} \cdot C,$$

with C the same constant as in the hypotheses, and where w is a multi-index. Technically, to obtain an estimate directly comparable with equation (8.19) in OpenAI’s NL paper, one would need also to take a supremum over all such all multi-indices w with $|w| \leq m$ ⁴, thereby obtaining

$$\|N^{-1} F\|_{C_y^m} \leq \hat{K}_m \|F\|_{C_y^{m+5}},$$

with $\hat{K}_m = \max(K_0, K_1, \dots, K_m)$. This estimate is then strictly weaker than equation (8.19) due to the extra derivative on the right-hand side.

Remark 3.2 (Consistent mismatch between $m + 4$ and $m + 5$ derivatives). Similar estimates as (3.2) can be seen throughout OpenAI’s Lean code: for example,

`inverse_finiteJets` and `mixedJet_inverse_bound`

each contain a similar $m + 5$ derivative requirement, rather than the expected $m + 4$ derivatives from the NL paper, as in (3.1).

The reason for the above discrepancy is related to the difference in proof method employed by the NL paper and the Lean code. As the quote above indicates, the NL paper makes use of the convergent series $\sum_{k \in \mathbb{Z}^2} \frac{1}{(1+|k|)^3}$ by using $m + 4$ derivatives. In contrast, the Lean code displayed above for

$$\text{norm_derivativeWord_inverse_le} \tag{3.3}$$

employs the series $\sum_{k \in \mathbb{Z}^2} \frac{1}{(1+|k_1|+|k_2|)^4}$ (see the exponent ‘4’ highlighted in red in the code (3.3) above) and thus uses $m + 5$ derivatives instead. Tracing the proof of (3.3) we find that the series arises from applying the Lean theorem `coefficient_seminorm_bound`, just as Figure 3 mentions. Consequently, the Lean results discussed in this section are weaker⁵ than (3.1) in the NL proof.

3.1.1. Explanation of the notation and technical information. The variable $y = (y_1, y_2)$ belongs to a two-dimensional torus $\mathbb{T}^2 = \mathbb{R}^2 / \mathbb{Z}^2$. The fixed vector and differential operator in Lemma 8.6 are

$$v_t = (\sqrt{2} - 1, 1), \quad N = v_t \cdot \partial_y = (\sqrt{2} - 1) \partial_{y_1} + \partial_{y_2}.$$

The subscript t is specified in equation (6.2) of the source paper. Here $F : \mathbb{T}^2 \rightarrow \mathbb{C}$ is smooth, with zero normalized Haar mean: $\int_{[0,1]^2} F(y) dy = 0$. The notation $N^{-1} F$ denotes the unique smooth zero-mean solution φ of $N\varphi = F$. The zero-mean requirement assures this solution is uniquely specified (else any solution, via addition of a constant, yields another). For smooth $G : \mathbb{T}^2 \rightarrow \mathbb{C}$ and a nonnegative integer r , the following norm is defined:

$$\|G\|_{C_y^r} := \max_{\substack{a, b \geq 0 \\ a+b \leq r}} \sup_{y \in \mathbb{T}^2} |\partial_{y_1}^a \partial_{y_2}^b G(y)|, \quad a, b \in \mathbb{N}.$$

³Strictly speaking, w in OpenAI’s Lean code is defined to be an object in the `derivativeWord` class: a list of booleans for which each ‘false’ entry corresponds to a derivative in the y_1 coordinate and each ‘true’ entry a derivative in the y_2 coordinate.

⁴This is not explicitly done as part of `norm.derivativeWord.inverse.le`.

⁵In fact, some such Lean results are actually incomparable with (3.1), due to the mismatch in the type of derivative. For example, the Lean theorem `inverse_finiteJets` uses mixed derivatives, not just torus derivatives. Note however that these results are still using $m + 5$ derivatives.

The subscript y means that other variables are held fixed. This is relevant in the case G is of the form $G(y) = F(p, y)$ for some $F : P \times \mathbb{T}^2 \rightarrow \mathbb{C}$ and some fixed p .

3.2. When a statement and its proof in the NL paper are both mistranslated into Lean. We now give an example that goes further than Example 2.2 in §2. In the following example the AI autoformalisation results in a Lean proof of a different bound obtained by a substantively different mathematical argument.

Example 3.3 (The NL paper’s pressure-flux bound and its proof are both mistranslated). Figure 4 concerns the estimate in equation (10.19) of OpenAI’s *Finite time blowup for Navier–Stokes*. We compare this estimate, together with its proof, with the cited Lean declaration at commit f9e8bc5. The issue here is not only a change in the bound, but also a change in the mathematical argument used to establish it. In particular, we have the following discrepancy: with $R \geq 1$, the comparison in Figure 4 is

$$\text{NL, equation (10.19): } \left| \int \pi w \cdot \nabla \chi_R \right| \leq \frac{C_T}{R} \left[(B_R + 1) B_R^{1/2} + R^{-3/4} B_R^{3/4} \right], \text{ for a.e. } t \in (0, T), \quad (3.4)$$

$$\text{Lean estimate: } \left| \int \pi w \cdot \nabla \chi_R \right| \leq C_T \left[(B_R^{1/2} + 1) \left(\frac{A_R}{R} + \frac{1}{R^2} \right) + R^{-7/4} B_R^{3/4} \right], \forall t \in (0, T), \quad (3.5)$$

where both the left hand side and the right hand side of both (3.4) and (3.5) depend on t ; see below for explanation of the notation. The NL paper’s bound is stated purely in terms of B_R , whereas the Lean estimate additionally depends on A_R . Here C_T is a finite nonnegative constant, allowed to depend on T (which itself is the maximum timescale for which the bounds are considered) and the norm conventions, but not on R . Although equation (10.17) in the NL paper bounds B_R above using A_R , it cannot be used to eliminate A_R from the Lean estimate to recover (10.19); the two bounds in (3.4) and (3.5) are therefore hard to compare.⁶

The closest Lean code to (10.19) (equation (3.4) in our paper) is the following result:

```
theorem exists_uniform_actual_pressure_flux_bound {T : ℝ} {u v : VelocityField}
  {p q : PressureField} (H : PressureRecovery.Hypotheses T u v p q)
  (M0 U0 G0 : ℝ) (hM0 : 0 ≤ M0) (hU0 : 0 ≤ U0) (hG0 : 0 ≤ G0)
  (hM : ∀ t ∈ Icc 0 T, MemLp (fun x => (u - v) (t, x)) 2 volume ∧
    comparisonLpNorm 2 (fun x => (u - v) (t, x)) ≤ M0)
  (hU : ∀ t ∈ Icc 0 T, MemLp (fun x => u (t, x)) 3 volume ∧
    comparisonLpNorm 3 (fun x => u (t, x)) ≤ U0)
  (hG : ∀ t ∈ Icc 0 T, ∀ i j : Fin 3,
    Integrable (tensorDiff u v t i j) volume ∧ comparisonLpNorm 1 (tensorDiff u v t i j) ≤ G0) :
  ∃ CP : ℝ, 0 ≤ CP ∧ ∀ R : ℝ, 1 ≤ R → ∀ t ∈ Ioo 0 T,
    | ∫ x, (p - q) (t, x) * fderiv ℝ (ComparisonCutoffs.weight R) x ((u - v) (t, x)) | ≤ CP *
      ((cutoffL6 (ComparisonCutoffs.cutoff R) (u - v) t ^ (1 / 2 : ℝ) + 1) *
        (dissipationRoot (ComparisonCutoffs.cutoff R) (u - v) t / R + 1 / R ^ 2) +
        R ^ (-7 / 4 : ℝ) * cutoffL6 (ComparisonCutoffs.cutoff R) (u - v) t ^ (3 / 4 : ℝ)) := by
  obtain ⟨CP, hCP, hbound⟩ := exists_uniform_canonicalCutoffFlux_bound M0 U0 G0 hM0 hU0 hG0
  refine ⟨CP, hCP, ?_⟩
  intro R hR t ht
  have ht' : t ∈ Icc 0 T := Ioo_subset_Icc_self ht
  have hu := NavierStokes.PeriodicUniqueness.spatial_smooth H.smooth_u ht'
  have hv := NavierStokes.PeriodicUniqueness.spatial_smooth H.smooth_v ht'
  exact (actual_flux_integrable_and_le_canonicalNorm H ht (zero_lt_one.trans_le hR) hu hv).2.trans
    (hbound R hR u v t hu hv (hM t ht').1 (hU t ht').1 (fun i j => (hG t ht' i j).1)
      (hM t ht').2 (hU t ht').2 (fun i j => (hG t ht' i j).2))
```

Source (GitHub permalinks): openai/NavierStokesAndEuler, commit f9e8bc5.

File: NavierStokes/R3/PressureFlux.lean.

Declaration: exists_uniform_actual_pressure_flux_bound, line 576.

Note that similar bounds (which are nonetheless distinct from (3.5)) appear in

exists_uniform_canonicalCutoffFlux_bound

⁶In fact, the purpose of the lemma is to establish that $w = 0$ and hence the (equal to 0) bound on the right hand side of (3.4) is stricter than the (non-zero) bound on the right hand side of (3.5). However, this is only established after completing the entire proof of Lemma 10.5.

Consider the natural language proof of the Navier Stokes problem recently announced by Open AI. We are concerned that the proof of the bound (10.19) has not been faithfully translated to the corresponding Lean formalisation. Point out any mistranslations or discrepancies you identify.

Worked for 3m 8s >

The Lean proof changes both the bound and its derivation.

1. A different bound replaces (10.19).

Using the paper's notation,

$$A_R = \left(\int \chi_R |\nabla w|^2 \right)^{1/2}, \quad B_R = \|\varphi_R^4 w\|_6,$$

the comparison is:

Paper, (10.19):

$$\left| \int \pi w \cdot \nabla \chi_R \right| \leq \frac{C_T}{R} \left[(B_R + 1) B_R^{1/2} + R^{-3/4} B_R^{3/4} \right].$$

[Paper, p. 123](#)

Lean, expressed in the same notation:

$$\left| \int \pi w \cdot \nabla \chi_R \right| \leq C_T \left[(B_R^{1/2} + 1) \left(\frac{A_R}{R} + \frac{1}{R^2} \right) + R^{-7/4} B_R^{3/4} \right].$$

This is the conclusion of [exists_uniform_actual_pressure_flux_bound](#).

The Lean estimate introduces dependence on A_R , losing the paper's bound in terms of B_R alone. Equation (10.17) bounds B_R above using A_R ; it cannot eliminate A_R from the Lean estimate to recover (10.19).

2. The proof method is mistranslated.

Paper: Commutes multiplication by φ_R^4 with $R_i R_j$. It estimates the local pressure term in $L^{3/2}$, pairs it with an L^3 factor, and uses interpolation to obtain $(B_R + 1) B_R^{1/2} / R$. [Paper, pp. 122–123](#)

Lean: Commutes multiplication by φ_R^2 . It pairs the weighted stress in $L^{6/5}$ with a Riesz-transformed test function in L^6 , then bounds that test function using a Sobolev estimate involving its derivative. This introduces A_R/R . See [fluxTest_eq_multiplier_rTest](#) and [riesz_rTest_bound](#) / [localized_pair_bound](#).

The final comparison proof [explicitly invokes this altered estimate](#).

📄 🗨️ ↩️ 🗑️

Do anything

+ 🗨️ Ask for approval

⚡ GPT-6 Astra Ultra 🔊 🗣️

FIGURE 4. ChatGPT-6 (Astra Ultra) shares our concern that the proof of the pressure-flux bound (10.19) in the announced proof of blow-up of Navier–Stokes is not semantically faithfully translated into Lean. Both the stated bound and the mathematical argument used to prove it differ from the cited Lean estimate. Example 3.3 explains this in further detail, including the notation.

and `canonicalCutoffFlux_bound`⁷. The bound (3.5) is given in the following Lean code, taken from that of `exists_uniform_actual_pressure_flux_bound`, Line 576.

```
|∫ x, (p - q) (t, x) * fderiv ℝ (ComparisonCutoffs.weight R) x ((u - v) (t, x))| ≤ CP *
((cutoffL6 (ComparisonCutoffs.cutoff R) (u - v) t ^ (1 / 2 : ℝ) + 1) *
(dissipationRoot (ComparisonCutoffs.cutoff R) (u - v) t / R + 1 / R ^ 2) +
R ^ (-(7 / 4 : ℝ)) * cutoffL6 (ComparisonCutoffs.cutoff R) (u - v) t ^ (3 / 4 : ℝ))
```

The discrepancy in (3.5) above arises through the diverging proof methods employed by the NL paper and the Lean code. It is beyond the scope, and besides the point, of this article to go into the full technical details of the proof technique for the NL paper's estimate (10.19) (in this paper, (3.4)). More information is provided below, but in summary three of the main reasons for the differences between the results and strategy used by the NL paper and the Lean argument are as follows:

- (1) The Lean proof does not rely on the boundedness of the Riesz transform as a map from $L^{3/2} \rightarrow L^{3/2}$; it instead invokes Sobolev embedding to move to L^2 and uses the boundedness of the Riesz transform in L^2 . The additional derivative involved in the Sobolev embedding is why (3.5) features A_R .
- (2) The Lean proof applies Hölder's inequality with different exponents, which changes the bounds at the end.
- (3) In the Lean proof, the left hand integral in (3.4) is explicitly shown to be continuous in time. This ensures that the statement for almost every $t \in (0, T)$ can be upgraded to a statement that avoids the 'almost every' qualifier.

3.2.1. Explanation of the notation and technical information. For context, inequality (10.19) appears itself in the proof of a velocity uniqueness result, namely Lemma 10.5. In the proof of Lemma 10.5, using the language of the NL paper, one chooses $0 \leq \varphi \leq 1$ to be smooth, compactly supported and equal to one on the unit ball. For $R \geq 1$, they write $\varphi_R(x) = \varphi(x/R)$ and $\chi_R = \varphi_R^8$. Continuing, the NL paper states: 'Let R_i be the Riesz transform with Fourier multiplier operator $i\xi_i/\|\xi\|_2$.' The NL proof of Lemma 10.5 begins by assuming that (v, P) and (u, p) are two solutions – with v (respectively u) the velocity and P (respectively p) the pressure – to a constructed Navier Stokes problem. The NL proof defines $w := v - u$ and $\pi := P - p$ and proceeds to establish the vanishing of w^8 ; the estimate (10.19) is a stepping stone towards this result. Note that v, u, w all depend on a time parameter t as well as spatial parameters; the time parameter is suppressed throughout the remaining discussion. The proof also defines a distribution $\pi_* = \sum_{i,j} R_i R_j g_{ij}$ where R_i, R_j are Riesz transforms, and where $g = (g_{ij})_{i,j \in \{1,2,3\}}$ with $g_{ij} = v_i v_j - u_i u_j = w_i w_j + u_i u_j + u_i w_j$. As indicated in Figure 4, using the NL paper's notation,

$$A_R := \left(\int \chi_R |\nabla w|^2 \right)^{1/2}, \quad B_R := \|\varphi_R^4 w\|_{L^6},$$

so that A_R is a localized L^2 gradient norm and B_R a localized L^6 velocity norm.

The purpose of defining π_* is that it is an attempt to get an explicit approximation for π in terms of the velocity fields; hence instead of directly analysing the integrand in (10.19), one replaces π in this term by π_* ⁹. The integrand is then factored into a product of two terms: $\varphi_R^4 \pi_*$ and another test function containing φ_R^3 . The Lean proof differs here, in that it splits the integral into $\varphi_R^2 \pi_*$ and another test function containing φ_R^5 ; we shall see the consequences of this change later.

⁷In addition, there is a similar argument that appears in the file `NavierStokes/R3StressPressureEstimate.lean` as part of the definition of `regularized_flux_Bound`. Once again the bound established there is different from the one in (3.4), and, intriguingly, does not appear to be used to prove the Navier Stokes theorem proposed by OpenAI. It is instead proven as one of the steps in the Lean compilation phase, but it is not directly used.

⁸The Lean proof actually uses $w = u - v$ and $\pi = p - P$, with the sign difference carried through in further definitions.

⁹A large part of the NL proof of Lemma 10.5 addresses this issue; ultimately, it suffices to show that $\nabla \pi = \nabla \pi_*$ in the sense of distributions.

The NL proof now tries to bound $\varphi_R^4 \pi_*$. The strategy here is to move φ_R^4 into the argument of the Riesz transformations in the definition of π_* ; doing so ensures that these Riesz transforms are being applied to functions with compact support in L^p spaces. In doing so one must pay a penalty, since $aR_i b \neq R_i a b$ in general for functions a, b . Hence (10.18) in the NL paper establishes that

$$\varphi_R^4 \pi_* = \sum_{i,j} R_i R_j (\varphi_R^4 g_{ij}) + \sum_{i,j} [\varphi_R^4, R_i R_j] g_{ij} \quad (3.6)$$

where $[\cdot, \cdot]$ is the standard commutator.

There are a number of differences in the approach the Lean argument uses here. As mentioned before, the proof in Lean takes out a factor of φ_R^2 instead of φ_R^4 . Furthermore, it does not consider π_* in the distributional sense, but rather directly considers its application on the relevant test function. Mathematically, the Lean argument is the following calculation

$$\begin{aligned} \int \pi_* w \cdot \nabla \chi_R &= \int \pi_* w \cdot 8 \varphi_R^7 \nabla \varphi_R = 8 \sum_{i,j} \int g_{ij} R_i R_j (w \cdot \varphi_R^7 \nabla \varphi_R) \\ &= 8 \sum_{i,j} \int \varphi_R^2 g_{ij} R_i R_j (w \cdot \varphi_R^5 \nabla \varphi_R) + 8 \sum_{i,j} \int g_{ij} [R_i R_j, \varphi_R^2] (w \cdot \varphi_R^5 \nabla \varphi_R). \end{aligned} \quad (3.7)$$

Both the NL proof and Lean proof then attempt to bound the first term in (3.6) and (3.7) respectively¹⁰. Here, there are substantial differences. The NL proof attempts to obtain an $L^{3/2}$ bound on the first term. It uses Hölder's inequality to show that $\varphi_R^4 g_{ij} \in L^{3/2}$ and the $L^{3/2} \rightarrow L^{3/2}$ boundedness of the Riesz transforms (which is established by citing [53]) to get a bound of the form $\|R_i R_j (\varphi_R^4 g_{ij})\|_{3/2} \leq C \|\varphi_R^4 g_{ij}\|_{3/2} \leq C' (\|\varphi_R^4 w\|_6 + 1) \|w\|_2 \leq C'' (B_R + 1)$, where C, C', C'' are all (potentially different) constants that can depend on T .

By contrast, the Lean proof does not rely on the above citation [53]. Instead, it uses a combination of Hölder's inequality (with Hölder conjugates 6 and 6/5), a Sobolev embedding theorem (bounding the L^6 norm by a multiple of the L^2 norm of the derivative, because this is an argument done over the spatial variables which are in three dimensions), an interchange of derivatives with the Riesz transform (which comes from the Fourier multipliers) and the boundedness of the Riesz operators from $L^2 \rightarrow L^2$ (which follows easily from considering the Fourier multipliers of these operators), with the following calculation

$$\begin{aligned} \left| \int \varphi_R^2 g_{ij} R_i R_j (w \cdot \varphi_R^5 \nabla \varphi_R) \right| &\leq \|\varphi_R^2 g_{ij}\|_{6/5} \|R_i R_j (w \cdot \varphi_R^5 \nabla \varphi_R)\|_6 \\ &\leq C \|\varphi_R^2 g_{ij}\|_{6/5} \|\nabla R_{ij} (w \cdot \varphi_R^5 \nabla \varphi_R)\|_2 \\ &= C \|\varphi_R^2 g_{ij}\|_{6/5} \|R_{ij} \nabla (w \cdot \varphi_R^5 \nabla \varphi_R)\|_2 \\ &\leq C \|\varphi_R^2 g_{ij}\|_{6/5} \|\nabla (w \cdot \varphi_R^5 \nabla \varphi_R)\|_2, \end{aligned}$$

where C is a constant from the Sobolev inequality. The Lean proof then bounds (using the product rule), for $x \in \mathbb{R}^3$ (and fixing the time parameter)

$$\begin{aligned} \|\nabla (w \cdot \varphi_R^5 \nabla \varphi_R)(x)\|_{\ell_2} &\leq \varphi_R^5(x) \|\nabla \varphi_R(x)\|_{\ell_2} \|\nabla w(x)\|_{op} + \varphi_R^5 \|\nabla^2 \varphi_R(x)\|_{op} \|w(x)\|_{\ell_2} \\ &\quad + 5 \varphi_R(x)^4 \|\nabla \varphi_R(x)\|_{\ell_2}^2 \|w(x)\|_{\ell_2}. \end{aligned}$$

Some further simplification is done (see for example `norm_fderiv_rle_amplitude`), but ultimately this expression leads to the $A_R/R + 1/R^2$ term present in (3.5). When $\varphi_R(x)$ was defined, it was done so that $\varphi_R(x) \in [0, 1]$, $\|\nabla \varphi_R(x)\|_{\ell_2} \leq C/R$ and $\|\nabla^2 \varphi_R(x)\|_{op} \leq C_2/R^2$. The A_R term comes from moving φ_R^4 inside the norm in $\|\nabla w(x)\|_{op}$.

¹⁰We do not comment here on how the second sum is treated by Lean and the NL, since differences in the bound and approach can easily be seen by observing the first sum.

Finally, we note that the $B_R^{1/2} + 1$ term comes from using the definition of $g_{ij} = w_i w_j + w_i u_j + u_i w_j$ to bound $\|\varphi_R^2 g_{ij}\|_{6/5}$. Indeed, the lean proof bounds both $\|\varphi_R^2 w_i w_j\|_{6/5}$ and $\|\varphi_R^2 u_i w_j\|_{6/5}$ separately as follows. Firstly, $\|\varphi_R^2 w_i w_j\|_{6/5} \leq \|\varphi_R w\|_{12/5}^2$ and then an interpolation inequality is used to get $\|\varphi w\|_{12/5} \leq \|w\|_2^{3/4} \|\varphi^4 w\|_6^{1/4} \leq C B_R^{1/4}$. Secondly, to bound $\|\varphi_R^2 u_i w_j\|_{6/5}$ a simple application of Hölder gives $\|\varphi_R^2 u_i w_j\|_{6/5} \leq \|u\|_3 \|w\|_2 \leq C$. This argument is also distinct from the NL proof, where no 12/5 interpolation is present in the proof of Lemma 10.5.

3.3. Mistranslations and OpenAI’s announced Euler proof. Alongside its Navier-Stokes announcement, OpenAI reported finite-time blow-up for the unforced incompressible Euler equations on $\mathbb{R}^3$, starting from smooth, compactly supported initial data. The Euler blow-up and well posedness question has a long history, and we will – as in the Navier-Stokes case – only mention a small subset of results here [2, 5, 12, 13, 16, 17, 19, 21, 34].

The format of the announced Euler paper is similar to the Navier-Stokes. There is a NL proof and an autoformalised Lean proof. As in the Navier-Stokes case, it is natural to ask whether this formalisation in Lean represents a semantically faithful translation of the NL paper, so that we can conclude that the NL paper represents a correct proof. It is beyond the scope of this paper to carefully examine the autoformalisation of the Euler proof in a similar way to what we have done in this paper with the Navier-Stokes case. However, based on some examinations of the autoformalisation of the Euler case – with both AI and manual tools – we think that the Euler proof will be another great case study for mistranslations in practice.

We encourage other readers to examine the autoformalisation of OpenAI’s Euler NL proof and determine if there are mistranslations. Our prediction is based on preliminary examinations, but our assessment is that the likelihood of substantial mistranslations is very high.

3.4. Why would mistranslations happen? When success is judged solely by obtaining a Lean proof which successfully compiles, the procedure followed by the AI can be summarised as follows.

Input: A faithful Lean translation X' of an NL theorem X and an alleged NL proof Y of X .
while no proof of X' has been verified **do**
 The AI revises Y' using Y as a guide (that is, the AI attempts a ‘translation’ of Y into Y')
 The AI checks if Y' is a valid Lean proof of X' .
end while
Output: the verified proof Y' .

The acceptance criterion tests whether Y' proves X' in Lean. It does not check whether the reasoning in Y has been preserved. A revision may repair an error or replace a correct argument by another one, just as in the above examples. Both count as success under this criterion, and neither establishes fidelity to the original purported proof. In fact, since the above procedure accepts any valid Lean proof, if the original NL proof is indeed incorrect, the AI is incentivised to find a different proof.

Remark 3.4 (Why AI cannot just translate the correct Lean proof back to English/LaTeX). It is natural to ask whether these mistranslation issues could be avoided by simply back-translating the produced Lean proof into NL and comparing with the original NL proof. It then remains, however, to establish both that the back-translation semantically faithfully represents the formal proof, and that its reasoning corresponds to the original argument. That is, back-translation requires further checks to ensure correspondence; it does not escape the objections raised in this section. In other words, we reach the same issue of semantic faithfulness not present in the outline for (i) but available only with autoformalisation in (ii) above, just going in the reverse direction. In summary, this approach is as difficult as type (ii).

4. LOGIC IN AI: WHY SEMANTIC PRESERVATION IN AI AUTOFORMALISATION IS HARD

For generations, logicians, mathematicians, linguists and philosophers have vigorously debated questions concerning natural and formal languages, semantics, and semantic preservation in translation [8, 27, 30, 38–

40, 43–45, 56, 59, 60]. For standard background on formal languages, interpretations and deductive systems, see [22, Chapters 1–2]. In autoformalisation semantic preservation becomes particularly delicate because of the need to resolve ambiguities, as we will see below. Throughout, we use $\mathbb{N} = \{0, 1, 2, \dots\}$, as in Lean’s `Nat`.

4.1. Logic in autoformalisation: Resolving ambiguities is harder than the Halting problem. A crucial problem facing any AI autoformaliser is the task of resolving ambiguities. Note that ambiguities may not always come from the fact that NLs produce ambiguous statements. Indeed, as the following example demonstrates, ambiguities can come from the fact that well-definedness of mathematical objects can depend on computational problems. This fact indicates that the hardness of resolving such an ambiguity is induced by the hardness of solving certain computational problems. Hence, results from foundations of mathematics & computation – such as the SCI hierarchy/arithmetical hierarchy – become important in order to understand the difficulty of resolving ambiguities.

Example 4.1 (A hidden existence presupposition causing ambiguities). An author of a textbook constructs explicitly a recursive enumeration (that is, there exists a computer program that can list them) $\{p_e\}_{e \in \mathbb{N}}$ of polynomials in $k + 1$ variables with integer coefficients. Later on, they write down a particular value of e , and define n_e to be the least natural number n for which $p_e(n, x_1, \dots, x_k) = 0$ has no solution in natural numbers $x_1, \dots, x_k$. They then define the real number $r_e = 1/(n_e + 1)$ and make the claim that it is obvious that $(r_e + 1)^2 = r_e^2 + 2r_e + 1$.

At first glance, this may seem like a trivial example to autoformalise. Indeed, the conclusion that $(r_e + 1)^2 = r_e^2 + 2r_e + 1$ is elementary to establish in Lean. The problem is the following: For Lean to prove that $(r_e + 1)^2 = r_e^2 + 2r_e + 1$, then r_e must be well-defined as a rational number. In particular, it must be established whether or not there exists an n such that $p_e(n, x_1, \dots, x_k) = 0$ has no natural-number solution $x_1, \dots, x_k$. This issue causes an ambiguity: can Example 4.1 be faithfully translated to Lean at all? The answer is as follows: yes, if r_e is well defined, and no if r_e is not well defined (when n_e does not exist). An AI autoformaliser aiming to translate Example 4.1 preserving the semantics is faced with the problem of resolving this ambiguity. If r_e is not well-defined, the AI autoformaliser should not attempt to translate in order to make Lean accept the translation.

Remark 4.2 (A semantic preserving AI autoformaliser must refrain from translating in certain cases). Note that if the AI autoformaliser attempts to ‘translate’ Example 4.1 when r_e is not well defined – and Lean accepts the ‘translation’ – the ‘translation’ cannot be semantically faithful, it must be a mistranslation. This begs the following question:

Can an AI autoformaliser resolve the ambiguity in Example 4.1 in general? That is, given $e \in \mathbb{N}$, can an AI autoformaliser determine when r_e is well defined?

The answer to this problem comes from Hilbert’s 10th problem [35] when $k \geq 9$:

No, such an AI autoformaliser can never exist, even if one could solve the Halting problem.

See Appendix A and Theorem A.3 for a detailed explanation of this statement. Example 4.1 shows why semantic preservation in mathematics is hard. In general, an AI cannot determine when a statement is ambiguous – and thus cannot be translated – and when a statement is not ambiguous and can be faithfully translated. Moreover, the above result can be strengthened and informally stated as follows:

Providing semantically faithful AI autoformalisation is harder than any computational problem.

See Appendix A.2.1 for a precise statement.

Example 4.3 (A natural attempt to resolve Example 4.1). It is natural to ask why one cannot simply apply the following algorithm, which may fail to halt and hence does not directly contradict the negative result concerning Example 4.1 above:

Input: Example 4.1 and $e \in \mathbb{N}$ (recall that e uniquely defines the claim in Example 4.1).

while no translation of Example 4.1 has been verified by Lean **do**

The AI revises an attempted translation Y' of Example 4.1 into Lean.

The AI checks if Y' is accepted by Lean.

end while

Output: the accepted translation Y' .

At face value, it seems like this methodology would only accept a translation when n_e is defined, and thus all choices of e for which the claim that $(r_e + 1)^2 = r_e^2 + 2r_e + 1$ in Example 4.1 is a true unambiguous statement would ultimately be resolved.

4.2. Why the algorithm in Example 4.3 cannot solve the problem. There is once again a problem with this approach. The AI could easily produce an attempted translation Y' which uses a Lean definition that supplies a default value, when n_e does not exist, in its proof that $(r_e + 1)^2 = r_e^2 + 2r_e + 1$. In particular, for a fixed $e \in \mathbb{N}$, write for $n \in \mathbb{N}$ the predicate $B_e(n)$ as follows:

$$B_e(n) \iff \forall (x_1, \dots, x_k) \in \mathbb{N}^k, \quad p_e(n, x_1, \dots, x_k) \neq 0. \quad (4.1)$$

With this predicate, a ‘translation’ of Example 4.1 in the case where $k = 2$ and

$$p_e(n, x_1, x_2) = x_1 + x_2 - n, \quad (4.2)$$

to Lean code (that compiles) can be done in the following way, with similar examples possible for a different choice of e and thus p_e . In particular, suppose an AI suggested the following ‘translation’ Y' of Example 4.1 into Lean:

```
import Mathlib

def p_e (n x1 x2 : Int) : Int := x1 + x2 - n

def B_e (n : Nat) : Prop :=
  forall (x1 x2 : Nat), p_e (n : Int) (x1 : Int) (x2 : Int) != 0

noncomputable def n_e : Nat := sInf {n : Nat | B_e n}

noncomputable def r_e : Rat := 1 / (n_e + 1)

theorem example_3_1 : (r_e + 1) ^ 2 = r_e ^ 2 + 2 * r_e + 1 := by ring
```

Here `sInf` returns the least element of a nonempty set of natural numbers, but returns 0 when the set is empty; see `Nat.sInf_def` and `Nat.sInf_empty` in [57]. Thus, if no natural number satisfies B_e , the code silently assigns $n_e = 0$ and $r_e = 1$. However, for the polynomial in (4.2), every $n \in \mathbb{N}$ admits the natural-number root $(x_1, x_2) = (n, 0)$, since $p_e(n, n, 0) = n + 0 - n = 0$. Hence, $B_e(n)$ is false for every n , and

$$\{n \in \mathbb{N} : B_e(n)\} = \emptyset.$$

In particular, r_e is not well-defined. The ‘proof’ succeeds without resolving the ambiguity and the existence presupposition because it mistranslates the statement by using `sInf`;

Warning: With the above ‘translation’ Y' (in the grey box) the algorithm in Example 4.3 will halt and produce a mistranslation. In this case the semantics of the definition of n_e have been completely changed. Indeed, if the above ‘translation’ into Lean had actually been semantically faithful, it would have implied that there is an $n \in \mathbb{N}$ such that $p_e(n, x_1, x_2) = x_1 + x_2 - n$ has no roots $(x_1, x_2) \in \mathbb{N}^2$ – this is false!

4.3. The root of the difficulty – Creating semantic preserving autoformalisers is a major open problem. All the problems discussed in the different examples above can be summarised in one rule of thumb, which illustrates the root of the problem:

Lean acceptance (compilation) of translation does not imply semantic preservation.

To illustrate the issue we consider the following example:

Example 4.4 (Lean compilation does not imply semantic preservation). Consider the following two sentences:

- (1) *There are infinitely many odd numbers among the natural numbers.*
- (2) *There are infinitely many prime numbers among the natural numbers.*

Changing the word ‘odd’ to ‘prime’ changes the semantics of the first sentence completely, as it becomes the second sentence. Suppose an AI should translate sentence (1) into Lean. Suppose also that it did so semantically faithfully and Lean compiles. What would have happened if the AI instead had made a mistake and changed ‘odd’ to ‘prime’, and then translated sentence (2) semantically faithfully to Lean – however, the intention was to translate sentence (1)? Lean would still compile, yet the ‘translation’ is actually a mistranslation.

Example 4.4 reiterates the message in §4.2: There are many candidate ‘translations’ that would make Lean compile – some of them would be semantically faithful translations – but many would be mistranslations. Thus, it will be up to the AI to choose a semantically faithful translation, Lean does not provide a security net if the AI provides a mistranslation. This means that the AI is subject to the limitations arising from the foundations of mathematics and computations as highlighted in §4.1 and Theorem A.3. Indeed, the AI autoformaliser cannot resolve all ambiguities and must therefore admit that there are many cases for which it will not know whether a translation is possible or not – even if a semantic translation is possible.

How to design trustworthy AI autoformalisers – meaning that they will only provide semantically faithful translations – is beyond the scope of this paper. Such developments will be announced in forthcoming articles. The key observation, however, is that even the most advanced models of OpenAI are far from achieving this goal. In particular, creating trustworthy AI autoformalisers that are capable of preserving semantics is a major open problem.

5. SUMMARY AND PERSPECTIVES

5.1. Attempting semantics-preserving autoformalisation in practice – The Meta attempt. Here is a pertinent example of the second type of autoformalisation – faithful semantic translation of a mathematical text – going awry in practice. During the spring of 2026, Meta declared a new milestone in autoformalisation: they claimed to have translated 26 mathematics textbooks into Lean, forming a library with 45,000 verified Lean declarations and around 500,000 lines of Lean code [46]. These claims were, however, swiftly refuted by the Lean community in the discussion quoted below [32].

- (i) Jack McCarthy: *‘Pg. 17 of the paper claims that 56% of the target statements in this book [i.e. one of the 26 books] were formalized, but I could not find a single one which did not contain a fatal error like above. The reality is that 0% of the statements here are accurately formalized.’*
- (ii) Brian Nugent: *‘I took a look at the algebraic geometry folder and it appears that the things it did prove (about 60% of the book according to your website) were exactly the things that are in Mathlib and it failed on everything else. [...] claiming “my AI autoformalized this AG textbook” in the state that’s its in is simply lying.’*
- (iii) Christian Merten: *‘I would like to push back a bit on the point that this project has produced something useful on this front.’*

Fortunately, in this case, Lean experts were quickly able to detect the mistakes made by the AI in the autoformalisation. This may not be the case in the future: proofs produced by autoformalisation are becoming longer and more complex. For example, Anthropic’s recent autoformalisation of Fermat’s Last Theorem took

13 million lines of Lean code [10]. The increases in both the number and length of autoformalised proofs will make it extremely difficult to catch mistranslations.

The problem of mistranslations by autoformalisation is particularly relevant in the context of building a unified body of knowledge in Lean, where results are consistent with, and build upon, one another. A single falsely translated statement in Mathlib means that all further results building upon this statement could be rendered invalid. Ensuring autoformalisation is done correctly is therefore of utmost importance to ensuring Lean libraries such as Mathlib remain correct and usable.

We may ask: *why did this autoformalisation project by Meta turn out the way it did?* The answer is that autoformalising AI systems currently follow something akin to the algorithm in Example 4.3 to produce translations. The project did go beyond this, checking for faithfulness and non-cheating in the translations; this semantic verification was, however, mainly performed by other AIs [46, Section 3 and Appendix D]. The fact that this failed highlights the difficulty that AIs have both with producing semantically faithful translations into Lean and with assessing the accuracy of such translations.

Remark 5.1. The crucial issue is that – as pointed out in Remark 4.2 – a semantic preserving AI formaliser must refrain from doing a translation in certain cases. The example in §4 and Theorem A.3 show that no AI autoformaliser can determine when it must refrain from translating and when it should translate. Thus, the big question is how to design AI autoformalisers that preserve semantics. Current AI based approaches are typically based on the ‘guessing and checking’ algorithm in Example 4.3, which is doomed to mistranslate because no AI can determine when it should or should not provide a suggested translation Y' (recall the notation from Example 4.3).

5.2. Semantics-preserving autoformalisation and the future of AI in mathematics. The Leiden Declaration from June 2026 [1] discusses how incorrect use of AI can be harmful to mathematics as a discipline. It singles out in particular the dangers posed by autoformalisation:

*‘Current automated techniques can produce plausible but unreliable (or even incorrect) arguments which are difficult to distinguish from correct mathematical proofs. **This applies not only to informal arguments, but also to formalizations, where the difficulty lies in the translation between computer-encoded and human presentations of concepts.***’

The boldface text refers exactly to the issues with the two forms of autoformalisation as discussed in this note. Even more recently, in September 2026, a declaration titled ‘A Severe Misalignment of AI in Mathematics’ [3] initiated by 25 Fields medalists brings up other issues related to AI in mathematics:

‘But solving problems is only a tool and proxy for achieving the primary goal of conceptual understanding and insight. Forgetting this in the world of AI may turn the tool against the primary goal. Indeed, the mass production at faster and faster pace of “true/false” statements could destroy fertile ground instead of breathing life into new ideas.’

The quote also relates fundamentally to autoformalisation:

- (i) Solving a problem only formally in Lean – without a semantically faithful translation from a correct NL argument that humans can read – would typically only be a proxy for the optimal final product which should provide understanding of the argument (both an NL argument and a Lean proof with a semantically faithful translation between the two). Without the semantically faithful translation, the NL argument could even be wrong – which is detrimental to human understanding – but would also harm future instalments of AI which rely on massive collections of correct mathematical NL literature in the training process.
- (ii) Pure ‘true/false’ statements coming from only Lean proofs (lacking a semantically faithful translation from a correct NL argument) can certainly be useful. Indeed, we would like to know if the Riemann Hypothesis is true. However, a principal reason for the interest in this problem stems from the belief that the techniques and ideas used in a proof could be applied broadly to address other questions and to

enhance our understanding across analysis, number theory and the many other fields that the Riemann Hypothesis touches on. A correct but incomprehensible Lean proof would not have the same value – mathematicians will unlikely read millions of lines of Lean code to try to understand. It would be even worse if an incorrect NL proof of the Riemann Hypothesis is accepted based on a mistranslated Lean proof; we would then have a complete misunderstanding of the facets of this problem.

As argued above, AI autoformalisers that can guarantee semantically faithful translations do not currently exist. Thus, developing semantically faithful AI autoformalisation stands as a major open problem, and attempts at quick fixes will typically violate fundamental limitations that date back to Gödel and Turing.

APPENDIX A. THE DIFFICULTY OF RESOLVING AMBIGUITIES AND HILBERT’S 10TH PROBLEM

A.1. Help from the SCI hierarchy and Hilbert’s 10th problem. Here we continue the presupposition resolution discussion from Example 4.1, explaining why no AI can resolve it for all $e \in \mathbb{N}$. The basic framework we use is that of the Solvability Complexity Index (SCI) hierarchy initiated in [28] and further developed in [4, 6, 7, 14, 15, 25]. The SCI hierarchy is closely connected to S. Smale’s [49, 50] program on the foundations of computational mathematics and scientific computing. This program motivated the early investigations of polynomial root-finding by C. McMullen [36, 37, 51] and P. Doyle & C. McMullen [20]. Their work yielded pioneering classification results within the SCI hierarchy, predating its formal introduction. See also the classification results in the SCI hierarchy by S. Weinberger [61]. Although some of the mathematics underlying the SCI hierarchy can be traced back to K. Gödel [26] and A. Turing [58], the newly developed techniques apply to any model of computation.

A.1.1. An informal introduction to the SCI hierarchy. Here we informally introduce the general SCI hierarchy (see [6, 14] for the formal definitions). The SCI hierarchy is designed to accommodate all types of computational problems and categorises the difficulty of computational problems according to the number of limits required to solve them. A computational problem is given by: a set of inputs Ω (where each element of Ω specifies a problem instance), a metric space $(\mathcal{M}, d)$ of candidate solutions, and a solution map $\Xi : \Omega \rightrightarrows \mathcal{M}$ that takes each problem instance to its solution(s) (the double arrow indicates that the solution could be multi-valued). There is in general also an evaluation set Λ specifying which information about the problem instances may actually be accessed by an algorithm. This will not be relevant¹¹ for this appendix. A computational problem is often specified simply as $\{\Xi, \Omega\}$, when $\mathcal{M}$ and Λ are understood.

The SCI of a computational problem is defined as the minimal number of limits¹² required to solve that problem ($+\infty$ if no finite number of limits suffices). The mainstay classes of the hierarchy are the Δ_m^α classes, which are defined for each $m \geq 0$. Here α is related to the model of computation, as explained below. Informally, we have the following description. Given a collection $\mathcal{C}$ of computational problems:

- (i) Δ_0^α is the set of problems that can be computed in finite time, so that they satisfy $\text{SCI} = 0$;
- (ii) Δ_1^α is the set of problems that can be computed using one limit (they satisfy $\text{SCI} \leq 1$) with control of the error, i.e., there is a sequence of algorithms $\{\Gamma_n\}$ with $d(\Gamma_n(A), \Xi(A)) \leq 2^{-n}$, $\forall A \in \Omega$;
- (iii) Δ_2^α is the set of problems that can be computed using one limit (again $\text{SCI} \leq 1$) but sans error control, i.e., there is a sequence of algorithms $\{\Gamma_n\}$ with $\lim_{n \rightarrow \infty} \Gamma_n(A) = \Xi(A)$, $\forall A \in \Omega$;
- (iv) Δ_{m+1}^α , for $m \geq 1$, is the set of problems that can be computed by using m limits, (in other words, $\text{SCI} \leq m$), i.e., there is a family of algorithms $\{\Gamma_{n_m, \dots, n_1}\}$ with

$$\lim_{n_m \rightarrow \infty} \dots \lim_{n_1 \rightarrow \infty} \Gamma_{n_m, \dots, n_1}(A) = \Xi(A), \forall A \in \Omega.$$

¹¹For the computational problems discussed in this appendix, each problem instance in Ω is specified entirely by a single natural number, so Λ amounts to containing only an identity map.

¹²This is formalised by the notion of towers of algorithms.

In general, this hierarchy can only be refined further, beyond these classes, if the metric space $\mathcal{M}$ admits additional structure, such as being totally ordered. On the other hand, the hierarchy typically does not collapse, so that we have:

$$\Delta_0^\alpha \subsetneq \Delta_1^\alpha \subsetneq \Delta_2^\alpha \subsetneq \dots \subsetneq \Delta_m^\alpha \subsetneq \dots \quad (\text{A.1})$$

Depending on the collection $\mathcal{C}$ of computational problems, however, the hierarchy (A.1) may terminate for a finite m or continue for arbitrary large m .

Definition A.1 (The SCI Hierarchy (totally ordered set)). Given the setup above, suppose in addition that $\mathcal{M}$ is a totally ordered set. Define

$$\Sigma_0^\alpha = \Pi_0^\alpha = \Delta_0^\alpha,$$

$$\Sigma_1^\alpha = \{\{\Xi, \Omega\} \in \Delta_2^\alpha \mid \exists \text{ a sequence of algorithms } \{\Gamma_n\} \text{ s.t. } \Gamma_n(A) \nearrow \Xi(A) \forall A \in \Omega\},$$

$$\Pi_1^\alpha = \{\{\Xi, \Omega\} \in \Delta_2^\alpha \mid \exists \text{ a sequence of algorithms } \{\Gamma_n\} \text{ s.t. } \Gamma_n(A) \searrow \Xi(A) \forall A \in \Omega\},$$

where $\nearrow$ and $\searrow$ denotes convergence from below and above respectively, as well as, for $m \in \mathbb{N}$,

$$\Sigma_{m+1}^\alpha = \{\{\Xi, \Omega\} \in \Delta_{m+2}^\alpha \mid \exists \text{ height } m+1 \text{ tower } \{\Gamma_{n_{m+1}, \dots, n_1}\} \text{ for } \{\Xi, \Omega\} \text{ s.t. } \Gamma_{n_{m+1}}(A) \nearrow \Xi(A) \forall A \in \Omega\},$$

$$\Pi_{m+1}^\alpha = \{\{\Xi, \Omega\} \in \Delta_{m+2}^\alpha \mid \exists \text{ height } m+1 \text{ tower } \{\Gamma_{n_{m+1}, \dots, n_1}\} \text{ for } \{\Xi, \Omega\} \text{ s.t. } \Gamma_{n_{m+1}}(A) \searrow \Xi(A) \forall A \in \Omega\}.$$

See [6] for the formal definition of the towers of algorithms used to define Σ_{m+1}^α and Π_{m+1}^α above. A specific case of the definitions of such towers, when the algorithms are Turing machines, is elaborated upon below. The above definitions apply in particular to the totally ordered metric space $\mathcal{M} = \{0, 1\}$; this yields the SCI hierarchy for arbitrary decision problems. Schematically, the general SCI hierarchy can be viewed in the following way:

$$\begin{array}{ccccccc} \Pi_0^\alpha & & \Pi_1^\alpha & & \Pi_2^\alpha & & \\ \parallel & \nearrow & \nearrow & \nearrow & \nearrow & \nearrow & \\ \Delta_0^\alpha & \subsetneq & \Delta_1^\alpha & \subsetneq & \Sigma_1^\alpha \cup \Pi_1^\alpha & \subsetneq & \Delta_2^\alpha \subsetneq \Sigma_2^\alpha \cup \Pi_2^\alpha \subsetneq \Delta_3^\alpha \subsetneq \dots \\ \parallel & \searrow & \searrow & \searrow & \searrow & \searrow & \\ \Sigma_0^\alpha & & \Sigma_1^\alpha & & \Sigma_2^\alpha & & \end{array} \quad (\text{A.2})$$

Note that the Σ_1^α and Π_1^α classes are crucial as they guarantee the existence of algorithms that will never make mistakes. They thus become crucial in computer-assisted proofs, see [6, Section 3].

We also use standard facts about the arithmetical hierarchy and oracle computability, for which we refer the reader to [52, Chapter 4]. The α in the Δ_m^α classes can refer to any model of computation (e.g. the classical Turing model [58] or the Blum-Shub-Smale model [9]). Hereon, we will take $\alpha = A$ where A refers to the arithmetic model of computation. This means the algorithms are all Turing machines i.e. the classical Turing model.

Remark A.2 (SCI hierarchy generalises the arithmetical hierarchy). We note that the arithmetical hierarchy becomes a special case of the SCI hierarchy [6, Proposition A.8]. More precisely, with $\alpha = A$ and $\mathcal{M} = \{0, 1\}$, if we restrict to computational problems asking about membership in a subset of $\mathbb{N}$, we have $\Sigma_m^A = \Sigma_m^0$, $\Pi_m^A = \Pi_m^0$ and $\Delta_m^A = \Delta_m^0$ for every $m \geq 0$ [6, Proposition A.8]. Here, Σ_m^0 , Π_m^0 and Δ_m^0 are the classical classification classes from the arithmetical hierarchy. We will make use of this fact in the following results.

Notation (Reducibility). For sets $A, B \subseteq \mathbb{N}$, we write $A \leq_m B$ if there is a total computable function $f : \mathbb{N} \rightarrow \mathbb{N}$ such that

$$(\forall n \in \mathbb{N})[n \in A \iff f(n) \in B].$$

In this case we say A is many-one reducible to B , witnessed by f . We write $A \leq_T B$ if there is an oracle Turing machine which, with B on the oracle tape, computes the characteristic function of A . In this case we say A is Turing reducible to B . The Turing degree of a set $A \subseteq \mathbb{N}$ is the set of all $C \subseteq \mathbb{N}$ that simultaneously satisfy $A \leq_T C$ and $C \leq_T A$. We say A is Σ_n^0 -complete if $A \in \Sigma_n^0$ and also $C \leq_m A$ for every $C \in \Sigma_n^0$.

Similarly, A is Π_n^0 -complete if $A \in \Pi_n^0$ and $C \leq_m A$ for every $C \in \Pi_n^0$. We fix an integer-coefficient polynomial pairing function $\langle \cdot, \cdot \rangle : \mathbb{N}^2 \rightarrow \mathbb{N}$ given by

$$\langle x, y \rangle = (x + y)(x + y + 1) + 2y, \text{ for } (x, y) \in \mathbb{N}^2,$$

which is a computable injection, by [41, Section 1, p. 1001]. By iteration, for each $n \geq 2$ we also obtain and hereon fix an n -ary pairing function $\langle \cdot, \dots, \cdot \rangle : \mathbb{N}^n \rightarrow \mathbb{N}$, which is also an integer-coefficient polynomial.

A.2. Resolving ambiguities is hard. Recall the condition defining n_e in Example 4.1. For an enumeration $(p_e)_{e \in \mathbb{N}}$ of integer-coefficient polynomials in the variables $n, x_1, \dots, x_k$, write

$$B_e(n) \iff (\forall (x_1, \dots, x_k) \in \mathbb{N}^k) p_e(n, x_1, \dots, x_k) \neq 0. \quad (\text{A.3})$$

Then n_e is defined to be the least $n \in \mathbb{N}$ for which $B_e(n)$ holds, provided such an n exists, and is undefined otherwise. Completeness throughout this subsection refers to completeness under many-one reductions.

Theorem A.3. Fix $k \geq 9$. For any computable enumeration $(p_e)_{e \in \mathbb{N}}$ of all polynomials with integer coefficients in $k + 1$ variables, let B_e be as in (A.3). Then the set

$$d := \{e \in \mathbb{N} : (\exists n \in \mathbb{N}) B_e(n)\}$$

is Σ_2^0 -complete. In particular, no AI can always determine whether the number n_e specified in Example 4.1 is well-defined, even with an oracle for the Halting problem.

Proof. We use the usual μ notation for minimum search [52]. By the assumption that the enumeration $e \mapsto p_e$ is computable and the assumption that p_e has integer coefficients, there exists an algorithm that decides if $p_e(n, x_1, \dots, x_k) = 0$ given inputs $n, x_1, \dots, x_k$. The condition defining membership in d is thus given by the quantifiers $(\exists n \in \mathbb{N})(\forall x_1 \in \mathbb{N}) \dots (\forall x_k \in \mathbb{N})$ over a decidable relation, so we have $d \in \Sigma_2^0$.

To obtain Σ_2^0 -completeness, fix a standard effective enumeration $(P_e)_{e \in \mathbb{N}}$ of all Turing machines. For $e, n \in \mathbb{N}$, write $P_e(n) \downarrow$ when $P_e(n)$ halts and $P_e(n) \uparrow$ when $P_e(n)$ does not halt. Recall that a Diophantine set [35] is a set $A \subseteq \mathbb{N}$ that can be represented using a finite number of unknowns, in the sense that there exists an integer $s \geq 1$ and an integer-coefficient polynomial $p(n, x_1, \dots, x_s)$ in $s + 1$ variables with:

$$n \in A \iff (\exists (x_1, \dots, x_s) \in \mathbb{N}^s) p(n, x_1, \dots, x_s) = 0.$$

The Davis–Putnam–Robinson–Matiyasevich (DPRM) theorem [35] states that the Diophantine sets are precisely the computably enumerable sets. The nine-unknown refinement [31] of the DPRM theorem permits every computably enumerable relation over $\mathbb{N}$ to be represented using nine unknowns. Applying this to the computably enumerable set $K_1 = \{\langle e, n \rangle \in \mathbb{N} : P_e(n) \downarrow\}$, we find there exists a fixed integer-coefficient polynomial $q_0(\tilde{n}, x_1, \dots, x_9)$, where $\tilde{n}$ is a variable, such that $\langle e, n \rangle \in K_1$ if and only if $q_0(\langle e, n \rangle, x_1, \dots, x_9)$ admits a root $(x_1, \dots, x_9) \in \mathbb{N}^9$. Defining the polynomial $q(e, n, x_1, \dots, x_9) := q_0(\langle e, n \rangle, x_1, \dots, x_9)$, we have for each $e, n \in \mathbb{N}$:

$$P_e(n) \downarrow \iff (\exists (x_1, \dots, x_9) \in \mathbb{N}^9) q(e, n, x_1, \dots, x_9) = 0.$$

For $k > 9$, regard q as independent of the additional variables $x_{10}, \dots, x_k$. Define the binary predicate $W \subseteq \mathbb{N}^2$ by

$$W(e, j) \iff p_j(n, x_1, \dots, x_k) \equiv q(e, n, x_1, \dots, x_k),$$

where $\equiv$ denotes that the two polynomials, whose variables are $n, x_1, \dots, x_k$, have coinciding coefficients; given e and j this is decidable. Since $(p_e)_{e \in \mathbb{N}}$ enumerates all integer-coefficient polynomials in $k + 1$ variables, the function $t(e) := \mu j[W(e, j)]$ is well-defined and computable. By the definitions of d and q , we have

$$t(e) \in d \iff (\exists n \in \mathbb{N}) P_e(n) \uparrow$$

In other words, t is a computable many-one reduction from

$$\text{NTOT} := \{e \in \mathbb{N} : (\exists n \in \mathbb{N}) P_e(n) \uparrow\}$$

to d . Since NTOT is Σ_2^0 -complete, so is d . By Post's theorem [52, Chapter 4], we have $d \leq_T \emptyset' \iff d \in \Delta_2^0$. Since $d \in \Sigma_2^0$, we have $d \leq_T \emptyset' \iff d \in \Pi_2^0$. But whenever a set belongs to Π_2^0 so too does any set that (many-one) reduces to it; thus if $d \in \Pi_2^0$ then by Σ_2^0 -completeness we would have $\Sigma_2^0 \subseteq \Pi_2^0$, contradicting strictness of the arithmetical hierarchy [52, Corollary 4.2.3, p. 85]. Consequently, we cannot have $d \leq_T \emptyset'$. Since d is exactly the set of e for which n_e in Example 4.1 is well-defined, the final claim follows. $\square$

The preceding result can be extended to arbitrarily high levels in the SCI hierarchy whilst simultaneously retaining polynomial predicates and the least-number construction of Example 4.1. We first generalise the predicate $B_e(n)$. For $l \geq 2$, consider an enumeration $(p_e)_{e \in \mathbb{N}}$ of integer-coefficient polynomials in the $l-1+k$ variables

$$n, z_2, \dots, z_{l-1}, x_1, \dots, x_k.$$

Let Q_j denote $\exists$ for odd j and $\forall$ for even j . We omit $z_2, \dots, z_{l-1}$ when $l = 2$. Define $B_e^{(l)}(n)$ by the following two cases. For even l , again omitting $z_2, \dots, z_{l-1}$ when $l = 2$, we have

$$\begin{aligned} B_e^{(l)}(n) &\iff (Q_2 z_2 \in \mathbb{N}) \cdots (Q_{l-1} z_{l-1} \in \mathbb{N}) (\forall (x_1, \dots, x_k) \in \mathbb{N}^k) \\ &\quad p_e(n, z_2, \dots, z_{l-1}, x_1, \dots, x_k) \neq 0, \end{aligned} \tag{A.4a}$$

For odd $l \geq 3$,

$$\begin{aligned} B_e^{(l)}(n) &\iff (Q_2 z_2 \in \mathbb{N}) \cdots (Q_{l-1} z_{l-1} \in \mathbb{N}) (\exists (x_1, \dots, x_k) \in \mathbb{N}^k) \\ &\quad p_e(n, z_2, \dots, z_{l-1}, x_1, \dots, x_k) = 0. \end{aligned} \tag{A.4b}$$

Note that (using the same k and polynomial enumeration) $B_e^{(2)}(n)$ is $B_e(n)$. As in [52], we write $\emptyset^{(0)} = \emptyset$ and $\emptyset^{(j+1)}$ for the Halting problem relative to $\emptyset^{(j)}$, for $j \in \mathbb{N}$.

Theorem A.4 (Presupposition resolution at arbitrary arithmetical levels). *Fix $l \geq 2$ and $k \geq 9$. Fix a computable enumeration $(p_e)_{e \in \mathbb{N}}$ of all integer-coefficient polynomials in $l-1+k$ variables, and let $B_e^{(l)}$ be as in (A.4). Then*

$$d_l := \{e \in \mathbb{N} : (\exists n \in \mathbb{N}) B_e^{(l)}(n)\}$$

is Σ_l^0 -complete. In particular, no algorithm can always determine whether the least n for which $B_e^{(l)}(n)$ holds is well-defined, even with an oracle for $\emptyset^{(l-1)}$.

Proof. Fix $l \geq 2$. By its definition, $B_e^{(l)}(n)$ is a Π_{l-1}^0 predicate. Indeed, it begins with $l-2$ alternating quantifiers - the first being a universal quantifier - and the subsequent k quantifiers are all either existential or all universal; the affected predicate therein, namely $p_e(n, z_2, \dots, z_{l-1}, x_1, \dots, x_k) = 0$, is decidable because the polynomial enumeration is computable and the computation is over the integers. That is, $d_l \in \Sigma_l^0$. For Σ_l^0 -hardness, let $A \subseteq \mathbb{N}$ be any Σ_l^0 set. By definition, there exists a computable relation $R \subseteq \mathbb{N}^{l+1}$ such that

$$a \in A \iff (Q_1 z_1 \in \mathbb{N}) \cdots (Q_l z_l \in \mathbb{N}) R(a, z_1, \dots, z_l). \tag{A.5}$$

Define a computably enumerable relation on the remaining parameters by

$$C(a, z_1, \dots, z_{l-1}) \iff \begin{cases} (\exists z_l \in \mathbb{N}) R(a, z_1, \dots, z_l), & \text{for } l \text{ odd;} \\ (\exists z_l \in \mathbb{N}) \neg R(a, z_1, \dots, z_l), & \text{for } l \text{ even.} \end{cases}$$

By the nine-unknown refinement of the DPRM theorem [31, 35] applied to the computably enumerable set $\{\langle a, z_1, \dots, z_{l-1} \rangle \in \mathbb{N} : C(a, z_1, \dots, z_{l-1})\}$, there exists a fixed integer-coefficient polynomial $q_0(\tilde{n}, x_1, \dots, x_9)$, where $\tilde{n}$ is a variable, such that

$$C(a, z_1, \dots, z_{l-1}) \iff (\exists (x_1, \dots, x_9) \in \mathbb{N}^9) q_0(\langle a, z_1, \dots, z_{l-1} \rangle, x_1, \dots, x_9) = 0.$$

Defining the polynomial $q(a, z_1, \dots, z_{l-1}, x_1, \dots, x_9) := q_0(\langle a, z_1, \dots, z_{l-1} \rangle, x_1, \dots, x_9)$, we have

$$C(a, z_1, \dots, z_{l-1}) \iff (\exists (x_1, \dots, x_9) \in \mathbb{N}^9) q(a, z_1, \dots, z_{l-1}, x_1, \dots, x_9) = 0.$$

If l is odd, this representation permits (A.5) to be expressed as

$$a \in A \iff (\exists z_1 \in \mathbb{N}) \cdots (\forall z_{l-1} \in \mathbb{N})(\exists(x_1, \dots, x_9) \in \mathbb{N}^9) \\ q(a, z_1, \dots, z_{l-1}, x_1, \dots, x_9) = 0.$$

If l is even, negation of the defining condition for C gives

$$(\forall z_l \in \mathbb{N}) R(a, z_1, \dots, z_l) \iff (\forall(x_1, \dots, x_9) \in \mathbb{N}^9) q(a, z_1, \dots, z_{l-1}, x_1, \dots, x_9) \neq 0,$$

so that (A.5) can be expressed as

$$a \in A \iff (\exists z_1 \in \mathbb{N})(\forall z_2 \in \mathbb{N}) \cdots (\exists z_{l-1} \in \mathbb{N})(\forall(x_1, \dots, x_9) \in \mathbb{N}^9) \\ q(a, z_1, \dots, z_{l-1}, x_1, \dots, x_9) \neq 0.$$

Consequently, regardless of the parity of l , we have replaced the final quantification ($Q_l z_l \in \mathbb{N}$) in (A.5) sans appending an additional quantifier alternation. When $k > 9$, we regard q as independent of the additional variables $x_{10}, \dots, x_k$. Define the binary predicate $W \subseteq \mathbb{N}^2$ by

$$W(a, j) \iff p_j(n, z_2, \dots, z_{l-1}, x_1, \dots, x_k) \equiv q(a, n, z_2, \dots, z_{l-1}, x_1, \dots, x_k),$$

where $\equiv$ denotes that the two polynomials, whose variables are $n, z_2, \dots, z_{l-1}, x_1, \dots, x_k$, have coinciding coefficients; given a and j this is decidable. Since $(p_e)_{e \in \mathbb{N}}$ enumerates all integer-coefficient polynomials in $l - 1 + k$ variables, the function $t(a) := \mu j[W(a, j)]$ is well-defined and computable. The preceding equivalences imply that

$$a \in A \iff t(a) \in d_l.$$

That is, $A \leq_m d_l$, establishing Σ_l^0 -completeness of d_l . By definition of d_l , the least $n \in \mathbb{N}$ satisfying $B_e^{(l)}(n)$ exists exactly when $e \in d_l$. By Post's theorem [52], we have $d_l \leq_T \emptyset^{(l-1)} \iff d_l \in \Delta_l^0$. Since $d_l \in \Sigma_l^0$, this becomes $d_l \leq_T \emptyset^{(l-1)} \iff d_l \in \Pi_l^0$. But whenever a set belongs to Π_l^0 so too does any set that (many-one) reduces to it; thus if $d_l \in \Pi_l^0$, then by Σ_l^0 -completeness we would have $\Sigma_l^0 \subseteq \Pi_l^0$, contrary to the strictness of the arithmetical hierarchy. Consequently, we cannot have $d_l \leq_T \emptyset^{(l-1)}$. $\square$

When we use the same k and the same polynomial enumeration, the case $l = 2$ gives $B_e^{(2)}(n) = B_e(n)$ for every $n \in \mathbb{N}$, so that $d_2 = d$, which recovers Theorem A.3. Moreover, for every $l \geq 2$, the complement of d_l is Π_l^0 -complete, and d_l has the Turing degree of $\emptyset^{(l)}$ (since $\emptyset^{(l)}$ is Σ_l^0 -complete by Post's Theorem [52]). For each fixed $l \geq 2$ and $e \in \mathbb{N}$, let $n_e^{(l)}$ be the least n for which $B_e^{(l)}(n)$ holds, and define $r_e^{(l)} = 1/(n_e^{(l)} + 1)$. Both $n_e^{(l)}$ and $r_e^{(l)}$ are left undefined if there is no such n . Replacing n_e and r_e in Example 4.1 by $n_e^{(l)}$ and $r_e^{(l)}$, respectively, gives the same least-number construction. The displayed identity $(r_e^{(l)} + 1)^2 = (r_e^{(l)})^2 + 2r_e^{(l)} + 1$ remains elementary to establish, while as e varies, deciding whether the stated definition of $r_e^{(l)}$ refers to a well-defined number is a Σ_l^0 -complete problem. Accordingly, we have established the existence of presupposition-resolution problems of unbounded finite arithmetical complexity.

For the SCI classification of the presupposition resolution problem we have thus far discussed, we use *arithmetical (computable) towers*; see [15, Definitions 5.2–5.6] for the formal definition. The input space Ω is $\mathbb{N}$ and the output space is $\mathcal{M} = \{0, 1\}$ with the discrete metric. A tower of height $h \geq 1$ for a function $\Xi : \Omega \rightarrow \{0, 1\}$ is given by a bottom-level computable function $\Gamma : \Omega \times \mathbb{N}^h \rightarrow \{0, 1\}$ that satisfies

$$\Xi(e) = \lim_{s_1 \rightarrow \infty} \cdots \lim_{s_h \rightarrow \infty} \Gamma(e, s_1, \dots, s_h),$$

for each $e \in \Omega$. In particular, each successive limit exists (for every fixed choice of the outer indices). An arithmetical tower of height zero for Ξ is simply a computable function $\Gamma : \Omega \rightarrow \{0, 1\}$ with $\Gamma = \Xi$. Given any function $\Xi : \Omega \rightarrow \{0, 1\}$, we define $\text{SCI}_A(\Xi)$ to be the least $h \geq 0$ for which there is an arithmetical tower of height h for Ξ ; we define $\text{SCI}_A(\Xi) = +\infty$ when no such tower exists. We also define Δ_{h+1}^A to denote the class of problems Ξ with $\text{SCI}_A(\Xi) \leq h$, for each $h \geq 1$.

Corollary A.5 (Unbounded SCI for presupposition resolution). *For each $l \geq 2$, define $\Xi_{d_l} : \Omega \rightarrow \{0, 1\}$ by $\Xi_{d_l}(e) = 1$ if and only if $e \in d_l$. With $k \geq 9$ and the enumeration in Theorem A.4, we have*

$$\text{SCI}_A(\Xi_{d_l}) = l \quad \text{for every } l \geq 2.$$

Equivalently, the decision problem for the set d_l lies in $\Delta_{l+1}^A \setminus \Delta_l^A$.

Proof. Fix $l \geq 2$. Since $d_l \in \Sigma_l^0$, we have $d_l \in \Sigma_l^A$ (as the SCI hierarchy generalises the arithmetical hierarchy, as discussed above). This means $\text{SCI}_A(\Xi_{d_l}) \leq l$.

On the other hand, if $\text{SCI}_A(\Xi_{d_l}) < l$, then there is an arithmetical tower of height $0 \leq h < l$ for Ξ_{d_l} . This would give that $d_l \in \Delta_{h+1}^0$ (again using that the SCI hierarchy generalises the arithmetical hierarchy) and thus also $d_l \in \Delta_l^0$ since $h < l$. But d_l is Σ_l^0 -complete so cannot lie in Δ_l^0 (else $\Sigma_l^0 \subseteq \Pi_l^0$, contrary to strictness of the arithmetical hierarchy [52, Corollary 4.2.3, p. 85]). This means $\text{SCI}_A(\Xi_{d_l}) \geq l$. We conclude that $\text{SCI}_A(\Xi_{d_l}) = l$, as required. $\square$

A.2.1. *The problem of resolving ambiguities has $\text{SCI} = \infty$.* Recall Example 4.1 and $B_e(n)$ from (4.1). Recall that the ambiguity in Example 4.1 – whether n_e is well defined or not – was caused by the following question:

‘Is $\{n \in \mathbb{N} : B_e(n)\}$ empty or not?’

Note that we could have made Example 4.1, and the definition of n_e , more general yielding an ambiguity caused by the question:

‘Is $\{n \in \mathbb{N} : B_e^{(l)}(n)\}$ empty or not?’

Here, $B_e^{(l)}(n)$ is defined for integers $l \geq 2$ in (A.4). Note that a trustworthy AI autoformaliser that will always provide an output: either a semantically faithful translation or ‘this statement cannot be translated’ must resolve the above question for each $e \in \mathbb{N}$ and $l \geq 2$. However, by Corollary A.5 – for each fixed l – this problem has $\text{SCI} = l$. Hence, the problem of answering the above question for all $e \in \mathbb{N}$ and $l \geq 2$ has $\text{SCI} = \infty$. We can now make sense of our informal statement:

Providing semantically faithful AI autoformalisation is harder than any computational problem.

By ‘semantically faithful AI autoformalisation’ we mean a trustworthy AI autoformaliser that will always provide an output – as above. Also, by ‘any computational problem’ we mean any computational problem with finite SCI. So our statement above – which was made informal for the purpose of readability – means: Every trustworthy semantically faithful AI formaliser (as above) must solve problems that are harder than any problem with finite SCI.

Remark A.6. The requirement that $k \geq 9$ represents the current state of literature on the DPRM theorem; this is an active area of research with a recent refinement [54] lowering the bound from $k \geq 11$ to $k \geq 10$ in the case where $x \in \mathbb{Z}^k$ (in our theorem and proof we work with $x \in \mathbb{N}^k$, where the best known bound for Σ_2^0 completeness of d is $k \geq 9$).

REFERENCES

- [1] J. Alper et al. Leiden Declaration on Artificial Intelligence and Mathematics, 2026. 2 June 2026. doi:10.5281/zenodo.20302944.
- [2] L. Alpöge and T. Buckmaster. Blowup for the Euler Equations with Smooth Forcing. Preprint, 2026.
- [3] A. Avila et al. A Severe Misalignment of AI in Mathematics, 2026. 11 September 2026. doi:10.5281/zenodo.22737750.
- [4] A. Bastounis, A. C. Hansen, and V. Vlačić. The extended Smale’s 9th problem – On computational barriers and paradoxes in estimation, regularisation, computer-assisted proofs and learning. *arXiv:2110.15734*, 2021.
- [5] J. T. Beale, T. Kato, and A. Majda. Remarks on the breakdown of smooth solutions for the 3-D Euler equations. *Communications in Mathematical Physics*, 94(1):61–66, 1984.
- [6] J. Ben-Artzi, M. J. Colbrook, A. C. Hansen, O. Nevanlinna, and M. Seidel. Computing spectra – On the solvability complexity index hierarchy and towers of algorithms. *arXiv:1508.03280v5*, 2020.
- [7] J. Ben-Artzi, A. C. Hansen, O. Nevanlinna, and M. Seidel. New barriers in complexity theory: On the solvability complexity index and the towers of algorithms. *Comptes Rendus Mathématique*, 353(10):931 – 936, 2015.
- [8] F. Berto and L. Hornischer. Cognitive synonymy: a dead parrot? *Philosophical Studies*, 180(9):2727–2752, 2023.

- [9] L. Blum, M. Shub, and S. Smale. On a theory of computation and complexity over the real numbers: NP-completeness, recursive functions and universal machines. *Bulletin of the American Mathematical Society (New Series)*, 21(1):1–46, 1989.
- [10] K. Buzzard. FLT: Anthropic has beaten me to it. Xena Project blog, 2026. 4 September 2026. Online article.
- [11] L. Caffarelli, R. Kohn, and L. Nirenberg. Partial regularity of suitable weak solutions of the Navier–Stokes equations. *Communications on Pure and Applied Mathematics*, 35(6):771–831, 1982.
- [12] J. Chen and T. Y. Hou. Stable nearly self-similar blowup of the 2D Boussinesq and 3D Euler equations with smooth data I: Analysis, 2022. Preprint; version 4, revised 16 August 2026.
- [13] J. Chen and T. Y. Hou. Stable Nearly Self-Similar Blowup of the 2D Boussinesq and 3D Euler Equations with Smooth Data II: Rigorous Numerics. *Multiscale Modeling & Simulation*, 23(1):25–130, 2025.
- [14] M. J. Colbrook, V. Antun, and A. C. Hansen. The difficulty of computing stable and accurate neural networks: On the barriers of deep learning and Smale’s 18th problem. *Proceedings of the National Academy of Sciences*, 119(12):e2107151119, Mar. 2022.
- [15] M. J. Colbrook and A. C. Hansen. The foundations of spectral computations via the Solvability Complexity Index hierarchy. *J. Eur. Math. Soc.*, 25(12):4639–4718, 2022.
- [16] P. Constantin, C. L. Fefferman, and A. J. Majda. Geometric constraints on potentially singular solutions for the 3-D Euler equations. *Communications in Partial Differential Equations*, 21(3–4):559–571, 1996.
- [17] D. Cordoba and L. Martinez-Zorua. Finite time singularities of smooth solutions for the 2D incompressible porous media (IPM) equation with a smooth source. arXiv:2410.22920 [math.AP], 2024. Preprint; version 3, revised 13 February 2025.
- [18] D. Cordoba, L. Martinez-Zorua, and F. Zheng. Finite time blow-up for the hypodissipative Navier Stokes equations with a force in $L_t^1 C_x^{1,\epsilon} \cap L_t^\infty L_x^2$. *Archive for Rational Mechanics and Analysis*, 250:38, 2026.
- [19] C. De Lellis and L. Székelyhidi, Jr. The Euler equations as a differential inclusion. *Annals of Mathematics*, 170(3):1417–1436, 2009.
- [20] P. Doyle and C. McMullen. Solving the quintic by iteration. *Acta Mathematica*, 163(3–4):151–180, 1989.
- [21] T. M. Elgindi. Finite-time singularity formation for $C^{1,\alpha}$ solutions to the incompressible Euler equations on $\mathbb{R}^3$. *Annals of Mathematics*, 194(3):647–727, 2021.
- [22] H. B. Enderton. *A Mathematical Introduction to Logic*. Academic Press, 2 edition, 2001.
- [23] C. L. Fefferman. Existence and smoothness of the Navier–Stokes equation. In J. Carlson, A. Jaffe, and A. Wiles, editors, *The Millennium Prize Problems*, pages 57–67. Clay Mathematics Institute and American Mathematical Society, 2006.
- [24] H. Fujita and T. Kato. On the Navier–Stokes initial value problem. I. *Archive for Rational Mechanics and Analysis*, 16(4):269–315, 1964.
- [25] L. E. Gazdag, A. Bastounis, and A. C. Hansen. Generalised hardness of approximation and the SCI hierarchy – on determining the boundaries of training algorithms in AI. *Foundations of Computational Mathematics*, July 2026.
- [26] K. Gödel. Über formal unentscheidbare Sätze der Principia Mathematica und verwandter Systeme I. *Monatshefte für mathematik und physik*, 38(1):173–198, 1931.
- [27] T. Hales. An argument for controlled natural languages in mathematics, 2019. Position essay, 20 June 2019.
- [28] A. C. Hansen. On the solvability complexity index, the n -pseudospectrum and approximations of spectra of operators. *Journal of the American Mathematical Society*, 24(1):81–124, 2011.
- [29] T. Y. Hou. Potentially Singular Behavior of the 3D Navier–Stokes Equations. *Foundations of Computational Mathematics*, 23:2251–2299, 2023.
- [30] T. F. Icard and L. S. Moss. A Simple Logic of Concepts. *Journal of Philosophical Logic*, 52:705–730, 2023.
- [31] J. P. Jones. Universal Diophantine equation. *The Journal of Symbolic Logic*, 47(3):549–571, 1982. doi:10.2307/2273588.
- [32] Lean community. Atlas (Meta). Zulip discussion in the Autoformalization channel, 2026. Discussion thread. Accessed 18 September 2026.
- [33] J. Leray. Sur le mouvement d’un liquide visqueux emplissant l’espace. *Acta Mathematica*, 63:193–248, 1934.
- [34] G. Luo and T. Y. Hou. Potentially singular solutions of the 3D axisymmetric Euler equations. *Proceedings of the National Academy of Sciences*, 111(36):12968–12973, 2014.
- [35] Y. V. Matiyasevich. *Hilbert’s Tenth Problem*. Foundations of Computing. MIT Press, Cambridge, MA, 1993.
- [36] C. McMullen. Families of rational maps and iterative root-finding algorithms. *Annals of Mathematics*, 125(3):467–493, 1987.
- [37] C. McMullen. Braiding of the attractor and the failure of iterative algorithms. *Inventiones Mathematicae*, 91(2):259–272, 1988.
- [38] R. Montague. Universal grammar. *Theoria*, 36(3):373–398, 1970.
- [39] R. Montague. English as a Formal Language. In R. H. Thomason, editor, *Formal Philosophy: Selected Papers of Richard Montague*, pages 188–221. Yale University Press, New Haven, CT, 1974. Originally published in 1970.
- [40] Y. N. Moschovakis. A logical calculus of meaning and synonymy. *Linguistics and Philosophy*, 29(1):27–89, 2006. doi:10.1007/s10988-005-6920-7.
- [41] M. B. Nathanson. Cantor polynomials and the Fueter–Pólya theorem. *The American Mathematical Monthly*, 123(10):1001–1012, 2016. doi:10.4169/amer.math.monthly.123.10.1001.
- [42] J. Nečas, M. Růžička, and V. Šverák. On Leray’s self-similar solutions of the Navier–Stokes equations. *Acta Mathematica*, 176(2):283–294, 1996.

- [43] W. V. O. Quine. Meaning and Translation. In R. A. Brower, editor, *On Translation*, pages 148–172. Harvard University Press, Cambridge, MA, 1959.
- [44] W. V. O. Quine. On the Reasons for Indeterminacy of Translation. *The Journal of Philosophy*, 67(6):178–183, 1970.
- [45] F. Rabe. How to identify, translate and combine logics? *Journal of Logic and Computation*, 27(6):1753–1798, 2017.
- [46] A. Rammal, N. Patel, F. Gloeckle, A. Hayat, J. Kempe, R. Munos, C. Arnal, and V. Cabannes. Formalizing mathematics at scale. [arXiv:2605.29955](https://arxiv.org/abs/2605.29955), 2026.
- [47] G. Seregin. A certain necessary condition of potential blow up for Navier–Stokes equations. *Communications in Mathematical Physics*, 312(3):833–845, 2012.
- [48] J. Serrin. On the interior regularity of weak solutions of the Navier–Stokes equations. *Archive for Rational Mechanics and Analysis*, 9:187–195, 1962.
- [49] S. Smale. The fundamental theorem of algebra and complexity theory. *Bulletin of the American Mathematical Society*, 4(1):1–36, 1981.
- [50] S. Smale. Complexity theory and numerical analysis. In *Acta numerica, 1997*, volume 6 of *Acta Numer.*, pages 523–551. Cambridge Univ. Press, Cambridge, 1997.
- [51] S. Smale. The work of Curtis T McMullen. In *Proceedings of the International Congress of Mathematicians I, Berlin*, Doc. Math. J. DMV, pages 127–132. 1998.
- [52] R. I. Soare. *Turing Computability: Theory and Applications*. Theory and Applications of Computability. Springer, Berlin–Heidelberg, 2016. doi:10.1007/978-3-642-31933-4.
- [53] E. M. Stein. *Singular Integrals and Differentiability Properties of Functions (PMS-30)*. Princeton University Press, 1970.
- [54] Z.-W. Sun. Ten unknowns for hilbert’s tenth problem over the integers, 2026.
- [55] T. Tao. Finite time blowup for an averaged three-dimensional Navier–Stokes equation. *Journal of the American Mathematical Society*, 29(3):601–674, 2016.
- [56] A. Tarski. The Semantic Conception of Truth: and the Foundations of Semantics. *Philosophy and Phenomenological Research*, 4(3):341–376, 1944.
- [57] The mathlib Community. Mathlib.Order.Lattice.Nat: Conditionally complete linear order structure on the natural numbers. Online API documentation. Documentation for `Nat.sInf_def` and `Nat.sInf_empty`. Accessed 18 September 2026.
- [58] A. M. Turing. On Computable Numbers, with an Application to the Entscheidungsproblem. *Proceedings of the London Mathematical Society*, S2-42(1):230, 1936.
- [59] J. van Benthem. Natural Language and Logic of Agency. *Journal of Logic, Language and Information*, 23(3):367–382, 2014.
- [60] J. van Benthem and F. Liu. New logical perspectives on monotonicity. In D. Deng, F. Liu, M. Liu, and D. Westerståhl, editors, *Monotonicity in Logic and Language*, pages 1–12, Berlin, Heidelberg, 2020. Springer Berlin Heidelberg.
- [61] S. Weinberger. *Computers, Rigidity, and Moduli: The Large-Scale Fractal Geometry of Riemannian Moduli Space*. Princeton University Press, USA, 2004.

APPENDIX B. FURTHER POTENTIAL MISTRANSLATIONS IN THE NAVIER–STOKES NL PROOF

B.1. AI generated list (with some human input) of suggested mistranslations. We compare OpenAI’s *Finite time blowup for Navier–Stokes* with its Lean sources at commit f9e8bc5. The twelve comparisons below are generated as a collaboration between ChatGPT and Claude, with some human input.

Warning – Potential hallucinations: Since the list below is generated with the help of AI, and has not been fully manually checked, hallucinations may occur. Hence, the list should be taken as a suggestion of mistranslations that could be checked by others. The purpose of the list is to give an indication of other potential mistranslations than the examples we provide in §3.

Notation and references. The singular time is $t = 1$. In cylindrical coordinates (r, θ, z) , r is distance from the axis and e_θ is the angular unit vector. The positive similarity scale q satisfies $q - z^2 q^{2h} = 1 - t$, where $0 < h < 1/100$; in particular, $q = 1 - t$ when $z = 0$. The profile coordinates are $X = r^2/(2q)$ and $\eta = z/q^{1/2-h}$; see (3.2), p. 7. The norm $\|\cdot\|_{C^m}$ controls all indicated derivatives through order m , and $O(q^\beta)$ means a bound by a constant times q^β as $q \downarrow 0$. Constants may depend on fixed derivative orders. Page numbers refer to the 166-page NL paper; Lean paths below are relative to `NavierStokes/`, and all line ranges link to the fixed commit.

1. Localization depends on the chosen potential

Paper: Theorem 3.1 represents the local velocity as $u_{\text{loc}} = \text{curl } A + D$, where A is a vector potential and $D = B e_\theta$ is a direct angular field. Equation (10.4) uses this pair to define $u_c = \text{curl}(cA) + cD$, with a smooth cutoff c .

Lean: The local-theorem pair is $A' = A_{\text{raw}} - J$, $D' = D_{\text{raw}} + \text{curl } J$, where J is the axial part removed from the base potential to make A' vanish in the exterior. The compact candidate instead cuts $A_{\text{raw}}, D_{\text{raw}}$. The theorem `velocity_eq_original` identifies the velocities before cutting.

Difference: For the same cutoff, the product rule gives

$$[\text{curl}(cA_{\text{raw}}) + cD_{\text{raw}}] - [\text{curl}(cA') + cD'] = \nabla c \times J.$$

This term does not vanish identically for the selected fields. At $z = 0$, the radial cutoff changes from one to zero for $s = r^2/2 \in (1/64, 1/32)$. At sufficiently late times, the entire segment from s to 1 lies in the heat exterior. There the axial coefficient of J is $-\int_1^s F_{\text{heat}}(v, t) dv$, which is positive because $F_{\text{heat}} > 0$ and $s < 1$. The radial gradient of the cutoff is nonzero somewhere on the transition and is perpendicular to J . Thus the selected localized velocities differ. Their agreement where $c = 1$ and the required properties of the compact candidate are separately proved.

Sources. Theorem 3.1, p. 15; (10.4), p. 118. `LocalPotentialRebundle.lean:24–30`,
`LocalPotentialRebundle.lean:96–109`; `LocalAngularGrowth.lean:138–147`;
`R3CompactCandidate.lean:201–202`; `TailGaugePotential.lean:180–195`, `TailGaugePotential.lean:228–323`;
`SpatialLocalization.lean:32–50`; `RadialHeatProfile.lean:121–122`;
`PhysicalHeatCoordinates.lean:39–44`; `PaperLocalization.lean:28–50`.

2. The cutoff violates the specified domain margin

Paper: The local fields are supplied on $\Omega_* = \{t < 1, q < q_*\}$. Proposition 10.1 chooses the spatial and temporal cutoffs so that $c \neq 0$ implies $q < q_*/2$. This leaves room inside the domain for a smooth extension by zero.

Lean: The cutoffs are fixed independently of q_* . Both equal one at $(t, x) = (3/4, 0)$. The displayed local threshold is

$$q_* = \min \left\{ Q_b, \frac{1}{2(|a_0| + 1)} \right\} \leq \frac{1}{2},$$

where $Q_b > 0$ is a selected band scale and a_0 is the first entry of the cutoff schedule. At that point $q = 1 - t = 1/4$, so $q < q_*/2$ fails.

Difference: This is a direct counterexample to the paper's cutoff-selection condition for the simultaneously displayed Lean choices. The formal construction uses raw fields smooth on all of $\mathbb{R}^3 \times \{t < 1\}$, allowing localization without that margin. The discrepancy is in the domain and extension argument; it does not exhibit a nonsmooth localized field.

Sources. §3.5, p. 16; Proposition 10.1, p. 118. `LocalPaperDomain.lean:16–17`; `SmoothCutoffs.lean:245–265`; `SpatialLocalization.lean:133–151`; `LocalScaleGeometry.lean:19–23`; `LocalPotentialRebundle.lean:78–94`.

3. A smooth axis value is replaced by totaled division

Paper: The auxiliary angular-stress quantity Q_s is defined for $X > 0$ by

$$Q_s(X, \eta) = \frac{\int_0^X H(s, \eta) S_q(s, \eta) ds}{XH(X, \eta)}, \quad Q_s(0, \eta) = \frac{1}{2} S_q(0, \eta).$$

Here $H = \sqrt{2X}E = 2XF$, E is the angular velocity profile, F is smooth and nonzero at the axis, and S_q is the smooth source in the radial equation (4.9). The second formula is the smooth extension required after (4.10).

Lean: `ProfileHistories.Profiles.angularLag` uses the quotient literally at every X . At $X = 0$, both numerator and denominator vanish, and Lean's division convention gives 0. The smooth source here is `NaturalEntrance.Sq`; the identity `naturalHistories.angularSource` identifies the integrand in `angularLag` with HS_q . For the selected entrance profile, `source_lower` gives $S_q(0, \eta) > 5/2$ for $|\eta| \leq 1$, so the smooth right-hand limiting value exceeds $5/4$.

Difference: For these entrance data the unweighted functions disagree at the axis, and the raw Lean quotient is discontinuous there. This does not imply discontinuity of the physical stress: Q_s enters it with a vanishing radial factor, and the original code proves the corresponding regular stress identities, including the axis. Regularity of those weighted quantities does not supply the missing value of the unweighted quotient.

Sources. (4.8)–(4.10), p. 26; (4.11), p. 27. `ProfileHistories.lean:554–588`; `NaturalEntrance.lean:1096–1105`, `NaturalEntrance.lean:1231–1267`; `ActivationStocks.lean:952–970`; `ZerothStressIdentity.lean:334–352`.

4. The pressure bump lies outside the reserved mean interval

Paper: Section 8.3 selects a smooth bump $\hat{\rho}$ with integral one and compact support in $I_m = \{\sqrt{2X} : X \in I_{\text{mean}}\}$. Here I_{mean} is a reserved subinterval of the active annulus; the bump removes the radial integral of the pressure source before compactly supported integration.

Lean: The actual pressure gauge supplies the full annulus endpoints $a = \sqrt{2X_a}$ and $b = \sqrt{2X_b}$ to `PressureStream.rho`. Its centre is $(a + b)/2$ and its outer radius is $(b - a)/4$, so the set on which it is nonzero is

$$\left(\frac{3a + b}{4}, \frac{a + 3b}{4} \right).$$

In the coordinate $\log X$, the reserved mean interval ends more than three units before the outer edge X_b . Passing to $\sqrt{2X}$, its upper endpoint is therefore less than $e^{-3/2}b < b/4$.

Difference: The selected bump is disjoint from the paper's reserved interval, since $(3a + b)/4 > b/4$. Its smoothness and unit integral are proved, and the separate stress-repair bumps use the reserved patch. This establishes a different auxiliary density; it alone does not establish different final pressure or velocity fields.

Sources. §8.3, p. 92. `PressureStream.lean:29–60`; `CorrectionInitialization.lean:4255–4259`; `VariableGaugeMean.lean:58–82`, `VariableGaugeMean.lean:517–523`; `CommonBaseContext.lean:118–125`; `PrimaryTargetBounds.lean:667–668`; `ReservedPatches.lean:29–64`; `NominalConeAssembly.lean:1327–1330`; `OutgoingSchedule.lean:304–306`; `OutgoingTail.lean:542–555`; `CorrectionInitialization.lean:4334–4342`.

5. Spatial phase derivatives pay the time-derivative cost

Paper: In Lemma 9.8, $Q \in (0, 1)$ is a fixed band scale comparable to q , Φ is an oscillatory phase, and $e^{ik\nu\Phi}$ is its carrier, where $k \leq 2Q^{-h/2}$ and the harmonic integer ν is fixed. For a spatial and b time derivatives, with $a + b \geq 1$, the proof retains the respective powers

$$Q^{-a/2-b(1+h)}, \quad Q^{-a(1/2+h/2)-b(1+3h/2)}.$$

These bounds also include constants and fixed powers of $1 + |\log Q|$.

Lean: `localPhase_positive_jets` bounds the full derivative tensor of order $a + b$ using $Q^{-(a+b)(1+h)}$. The carrier and its multiplication by an amplitude are controlled using $Q^{-(a+b)(1+3h/2)}$; the same cost is charged in every direction.

Difference: For $a > 0$, both comparisons lose an additional factor $Q^{-a(1/2+h)}$. These are weaker intermediate bounds, rather than the directional estimates displayed in the paper's proof. Nevertheless, the paper subsequently uses the common loss $\ell_m = 2A + (m + 1)(1 + 3h/2)$, where $A = 1/2 + h$. Lean proves this final loss exactly in `paper_physical_loss`; the discrepancy does not concern the conclusion of Lemma 9.8.

Sources. Lemma 9.8 and its proof, pp. 113–114. `SharpCurrentPhysicalPhase.lean:168–175`, `SharpCurrentPhysicalPhase.lean:283–290`; `SharpCurrentModeJets.lean:18–35`; `SharpPhysicalCarrier.lean:12–17`; `WholeDomainPhysicalStageTheorem.lean:238–264`.

6. Frozen scale jets and Euler derivatives give equivalent seminorms

Paper: The normalized profile estimates (5.42) and (5.46) use compositions of $q\partial_q$, ∂_X and ∂_η . Repeating $q\partial_q$ differentiates its coefficient q as well as the profile. The diagonal bound in (5.46) is imposed for $n \geq \max\{2, |I|\}$, where $|I|$ is the derivative order.

Lean: `SlowBorelBase.blownJet` takes ordinary derivatives of $F(q\sigma, X, \eta)$ at $\sigma = 1$, keeping the base scale q fixed. For the pure scale directions, this means

$$\partial_\sigma^2 F(q\sigma, X, \eta)|_{\sigma=1} = q^2 \partial_q^2 F, \quad (q\partial_q)^2 F = q^2 \partial_q^2 F + q\partial_q F.$$

The `AdmissibleScales.blown` condition bounds the tensor of order m of the n th summand by $2^{-n} q^{hn}$ for $n \geq 1$, $m \leq n + 2$, $0 < q \leq 1$, and (X, η) in the chosen compact profile set. The theorem `exists_admissibleScales` constructs a schedule satisfying these conditions.

Difference: The operators are unequal already for $F(q) = q$: the displayed second derivatives are 0 and q . This example compares operators, not the selected profiles. At every fixed order the Euler and frozen derivative families are related by an invertible triangular change of basis with constant coefficients. The same holds for mixed profile derivatives. Consequently they give equivalent all-order $O(q^{2h})$ estimates. Transferring the displayed diagonal bound can introduce an order-dependent multiplicative constant; it does not automatically retain the exact coefficient 2^{-n} . No failure of either the paper's asymptotic estimates or its numerical bound is established here.

Sources. (5.42), p. 60; (5.46), p. 61. `SlowBorelBase.lean:136–139`, `SlowBorelBase.lean:289–355`, `SlowBorelBase.lean:560–566`.

7. The cone margin is chosen on a smaller family

Paper: Appendix C constructs a periodic shear loop with slope $\vartheta(\theta'; \mu)$, mean slope m , and amplitude $\mu \geq 0$. Write $V(\mu) = \langle (\vartheta - m)^2 \rangle$ for its variance, where $\langle \cdot \rangle$ averages over $0 \leq \theta' \leq 2\pi$. The admissible speed satisfies $2 < v < U(\vartheta)$, with U the upper boundary of the stress cone. After choosing $\mu_{\max}$, (C.8) requires

$$U(\vartheta(\theta'; \mu)) > 2 + \delta_L \quad \text{for every } 0 \leq \mu \leq \mu_{\max}, \quad 0 < \delta_L < 1.$$

Lean: `FamilyChoices.cone_margin` instead quantifies over solved tilts with $s = \sqrt{V} \leq \sqrt{3/a_{\min}}$, where $a_{\min} > 0$ is a uniform lower bound for the prescribed angular shear a . Its simultaneously selected `maxAmplitude` satisfies $V(\mu_{\max}) > 3/a_{\min}$, so that endpoint lies outside this family. The selected inequalities are $U \geq 2 + \delta$ and $0 < \delta \leq 1$.

Difference: The selected family is smaller than the full amplitude interval associated with `maxAmplitude`. Its margin suffices for the actual loop, whose strict admissibility is proved. The non-strict auxiliary inequalities are harmless: replacing δ by $\delta/2$ gives a strictly smaller positive margin less than one on the same family. A generic compact-family theorem can also provide a margin on the larger interval, but that is not the selected application. This is a difference in auxiliary choices, not a counterexample to the paper's margin or to the loop construction.

Sources. (C.4)–(C.8), p. 159; (C.9)–(C.10), p. 160. `TrueConeLoop.lean:164–197`, `TrueConeLoop.lean:371–393`, `TrueConeLoop.lean:439–504`; `LoopVariance.lean:842–854`.

8. Uniform full-pressure bounds omit decay of the pressure increment

Paper: Fix the outer and axis data and a sufficiently large reference scale Λ . Let C be the subsequent sufficiently large admissible amplitude parameter, φ_r the normalized reference angular profile, and $\Pi_0(\eta)$ the axis pressure. With $\eta \in [-1, 1]$ and the reference transition width in its prescribed range, the pressure satisfies

$$\Pi_r(X, \eta) - \Pi_0(\eta) = C^{-2} \int_0^X \varphi_r(s, \eta)^2 ds, \quad \sup_{0 \leq X \leq X_i} \|\Pi_r - \Pi_0\|_{C_\eta^k} \leq K_k C^{-2}.$$

Here $X_i = 110$ is fixed and K_k is independent of admissible C and the transition width. Thus (B.24) retains smallness as $C \rightarrow \infty$, at every fixed order.

Lean: `uniform_refF_normalized` controls the η derivatives of Cf , where $f = \varphi_r/C$. The auxiliary theorem `uniform_refF` instead bounds C^{-1} by 1. In `uniform_ref_histories`, this coarser estimate is squared and integrated, and the fixed axis pressure is added. The resulting theorem controls the *full pressure* uniformly in C ; it does not display (B.24)’s C^{-2} bound for the *increment* $\Pi_r - \Pi_0$. Its indicated consumer uses uniform full-pressure control.

Difference: The selected intermediate bound omits the smallness of the pressure increment; a bound for the full pressure is not expected to decay when Π_0 does not. Sharp order-zero and first-parameter-derivative increment bounds are separately proved and used. At every higher fixed order, the normalized-field bounds, the product rule and the pressure integral formula also imply a C^{-2} increment bound. Thus this is a difference in the estimate displayed and used at this step, not evidence that (B.24) fails for the formalized profiles.

Sources. Lemma B.4 and (B.24), pp. 150–151. `ReferenceUniformStocks.lean:245–279`,
`ReferenceUniformStocks.lean:285–310`, `ReferenceUniformStocks.lean:341–352`;
`ReferenceBounds.lean:96–139`, `ReferenceBounds.lean:955–987`.

9. A full-tensor estimate gives a coarser mixed-derivative bound

Paper: Lemma 10.3 extends the force beyond $t = 1$ by summing $G_j(x, \sigma) = \chi_0(b_j \sigma) \sigma^j F_j(x)/j!$, where $\sigma = t - 1$, F_j is the limiting j th time derivative of the force, χ_0 is a fixed smooth cutoff, and $b_j \geq 1$ controls its width. For a spatial multi-index α and m time derivatives, (10.12) gives

$$\|\partial_x^\alpha \partial_\sigma^m G_j\|_\infty \leq C_{j,m} b_j^{m-j} \|F_j\|_{C^{|\alpha|}}.$$

Only time derivatives differentiate the rescaled cutoff.

Lean: `SpatialBorelExtension.localizedTermDerivBound` estimates the full derivative tensor of total order k on a spatial window by $b_j^{k-j} C_{j,k,W}$. Here $C_{j,k,W}$ is a bound for a fixed template built from F_j on the window W , independent of b_j . Setting $k = |\alpha| + m$ therefore incurs $b_j^{|\alpha|+m-j}$. The proof bounds the norm of the time-rescaling map in all directions.

Difference: Specializing the full-tensor bound to a mixed component gives a coarser scale factor, with an additional $b_j^{|\alpha|}$. This is a cost of the norm estimate, not an actual spatial derivative of the time cutoff. The template constant controls derivatives of F_j up to k , whereas the paper’s mixed bound uses only order $|\alpha|$. The pure-time theorem retains b_j^{m-j} , and the coarser joint estimate still proves convergence with every derivative. Spatial windows are used for bounds; their cutoffs do not appear in the final extension series.

Sources. Lemma 10.3, (10.11)–(10.12), p. 120. `SpatialBorelExtension.lean:63–69`,
`SpatialBorelExtension.lean:105–174`, `SpatialBorelExtension.lean:270–304`,
`SpatialBorelExtension.lean:334–388`; `BorelExtension.lean:125–155`.

10. A slower envelope controls the entire future tail

Paper: During the axial transition, the pressure argument uses decay $E(y + s, \eta) \leq K E(y, \eta) e^{-s/2}$ for every $s \geq 0$. Here $y = \log(X/X_R)$ is the logarithmic X -coordinate, with fixed $X_R > 0$, s is an increment in y , and E is the positive unmodified outgoing angular profile. Squaring and integrating this envelope controls the future pressure integral $\Pi(y, \eta) = -\frac{1}{2} \int_y^\infty E(v, \eta)^2 dv$.

Lean: For the actual profile at $\eta = 0$, `logClock_future` and `finalAngular_clock_ratio` give

$$E(y + s, 0) \leq e^{3/5} E(y, 0) e^{-2s/5} \quad (y, s \geq 0).$$

The squared decay rate is $4/5$. Comparisons between $E(y, \eta)^2$ and $E(y, 0)^2$ then yield

$$\int_y^\infty E(v, \eta)^2 dv \leq 5e^{6/5} E(y, \eta)^2 \quad (y \geq 0, |\eta| \leq 1).$$

Difference: The selected proof uses the slower rate $2/5$ instead of $1/2$, and transfers a parameter-zero envelope to the physical parameter range. The resulting bound is used for the actual pressure. The original `finalAngular_release_decay` proves the sharper rate $1/2 + 3h/4$ after the release stage; its later starting point does not replace the all-future argument used here. No failure of the pressure bound or of tail integrability follows.

Sources. (A.23), p. 133; axial-transition pressure estimate, p. 135; exterior decay, p. 137. `FuturePressureBounds.lean`:106–160, `FuturePressureBounds.lean`:171–217; `OutgoingEntranceCone.lean`:877–889; `TailCone.lean`:227–243.

11. The local hierarchy is completed before global repair

Paper: Section 5 builds a formal expansion in powers q^{2nh} , $n \geq 0$. Its coefficients are profiles of the velocity and pressure in (X, η) . At order n , Lemma 5.1 solves the equations near the axis using the already finalized lower orders. Lemma 5.2 then extends the current profiles and imposes five integral moment conditions, which confine their stress to an annulus. Only then does the induction proceed to order $n + 1$.

Lean: `SlowRecursion.hierarchy` first recursively constructs every local coefficient, without global moment repair. `schemeFromHierarchy` cuts these local coefficients to produce the seeds for a second recursion. In that global recursion, `exists_step` repairs the current seed and recomputes its pressure source and moments using the already repaired global predecessors.

Difference: The code separates the local and global inductions, whereas the paper interleaves them. The second recursion takes a completed local hierarchy as input. The theorem `profiles_inner_eq` proves agreement of all four velocity/pressure profile components on a common inner region, including the axis, and `profiles_moments` proves the global moment conditions. These are the connecting results needed for the separated construction. This demonstrates a reorganization of the construction and its verification; it does not by itself demonstrate that the resulting coefficients differ from an admissible implementation of the paper's induction.

Sources. §5, p. 45; Lemma 5.1, p. 47; Lemma 5.2, p. 50; Proposition 5.3, p. 55. `SlowRecursion.lean`:931–972; `GlobalSlowProfiles.lean`:770–834, `GlobalSlowProfiles.lean`:1043–1052, `GlobalSlowProfiles.lean`:1090–1118, `GlobalSlowProfiles.lean`:1812–1825.

12. Affine bumps and Rolle's theorem replace geometric dilates

Paper: Lemma 8.7 repairs five moments on an interior radial interval. In the normalized radius $x = r/\sqrt{q}$, the angular moment weights are $x^2, x^{-2-2\lambda}, x^{-2\lambda}$, and the axial weights are $x, x^{1-2\lambda}$, with $\lambda > 0$ fixed. The proof chooses geometric dilates $\beta_j(x) = e^{-jd}\beta_0(e^{-jd}x)$, $j = 0, 1, 2$, of one nonzero nonnegative bump, with $d > 0$. Their moments are $\int x^p \beta_j(x) dx = \mu_p e^{jdp}$, where $\mu_p > 0$. Dividing rows by μ_p gives ordinary Vandermonde matrices. The axial block reuses the first two angular bumps.

Lean: `FiveRowRank` uses affine copies on evenly spaced cells inside (a, b) : for a family of size N , the cell endpoints are $a + (2j + 1)\Delta_N$ and $a + (2j + 2)\Delta_N$, with $\Delta_N = (b - a)/(2N + 1)$ and $0 \leq j < N$. Thus the angular family uses $N = 3$, while the axial family uses the different mesh $N = 2$. Nonsingularity follows by turning a vanishing moment combination into one zero in each cell, then applying Rolle induction to sums of distinct powers after logarithmic change of variable.

Difference: The selected bump families and the proof of invertibility differ from those in Lemma 8.7; in particular, the axial bumps are not the first two angular bumps. However, the paper itself uses Rolle induction for general separated power-moment bumps in Lemma A.1. Thus the formal argument is closely related to another argument already in the paper. The five required moment equations, including their signs, are proved. This is an alternative realization of the moment repair, not missing constraints.

Sources. Lemma 8.7, pp. 97–98; Lemma A.1, pp. 126–127. `FiveRowRank.lean:24–31`, `FiveRowRank.lean:81–107`, `FiveRowRank.lean:146–156`, `FiveRowRank.lean:240–284`; `LocalizedMomentRepair.lean:21–26`, `LocalizedMomentRepair.lean:87–115`; `PowerMomentMatrix.lean:50–117`, `PowerMomentMatrix.lean:209–249`.

DEPARTMENT OF MATHEMATICS, KING'S COLLEGE LONDON

Email address: alexander.bastounis@kcl.ac.uk

DEPARTMENT OF APPLIED MATHEMATICS AND THEORETICAL PHYSICS, UNIVERSITY OF CAMBRIDGE

Email address: fac43@cam.ac.uk

DEPARTMENT OF APPLIED MATHEMATICS AND THEORETICAL PHYSICS, UNIVERSITY OF CAMBRIDGE

Email address: a.hansen@damtp.cam.ac.uk