
Logic and Foundations in AI: On Undecidability, Instability, and the Automated Mathematician

ANDERS HANSEN^{1,2} and FABIAN CIRCELLI¹

¹*University of Cambridge*

²*University of Oslo*

Short editorial introduction

Abstract The early days of AI – initiated by Turing – were dominated by logic and the foundations of computing. On the other hand, AI developments over the last decade have been largely engineering-driven based on a trial-and-error methodology using neural networks. While the success of this approach is undisputed, concerns about trustworthiness in AI have arisen from “hallucinations” (false but plausible information) and instabilities occurring widely. Recently, there has been a revival of logic and foundations in AI in order to address these issues. In particular, by going back to the roots of AI – based on the legacy of Turing and his use of logic – one can deal with these matters from a theoretical point of view. Finally, we consider the question of whether a program on logic in AI for trustworthiness can result in a trustworthy scientific computing chatbot – that never produces a wrong output – which can replace computational mathematicians.

1 Introduction

This paper highlights the essential role that logic and foundations of mathematics play in the development of trustworthy AI. Despite their undisputed capabilities and continuous improvements, AI systems remain plagued with hallucinations (false but plausible information), among other errors [17, 18, 20, 22]. In fact, the Cambridge Dictionary selected “hallucinate” as its 2023 Word of the Year [14], underscoring the impact that AI hallucinations are having on global discourse. As AI is increasingly incorporated in safety-critical and high-stakes applications – including medicine [34] and security [26] – ensuring the trustworthiness of AI outputs is essential. Key questions of interest are:

- Q: Why are hallucinations rampant in AI systems? To what extent can they be remedied?
- Q: Why are so many trained neural networks unstable when stable neural networks provably exist?
- Q: Which computational problems can be correctly solved by AI systems?

We demonstrate how logic and foundations of mathematics are inextricably linked to answering the above questions, and are therefore necessary tools in developing trustworthy AI. We highlight the need for a program on AI that is based on logic and computability theory and cover some work done in this direction.

2 Undecidability in imaging

In this section, we demonstrate how non-computability [44, 46] phenomena manifest in imaging applications when attempting to compute neural networks in, for example, magnetic resonance imaging (MRI). The primary inverse problem of interest in imaging is to recover $x \in \mathbb{C}^N$, given a measurement $y \in \mathbb{C}^m$ and a matrix $A \in \mathbb{C}^{m \times N}$ with $y = Ax$. A key issue in imaging is accelerated acquisition (techniques to reduce scan times), because it results in an undersampled inverse problem with $m < N$. A classical technique to deal with undersampling is to regularise the undersampled inverse problem, for example using

LASSO [1, 28]. LASSO concerns the problem of finding $\hat{x}$ satisfying

$$\hat{x} \in \operatorname{argmin}_{x \in \mathbb{C}^N} \|Ax - y\|_2^2 + \lambda \|x\|_{\ell^1}, \quad (1.1)$$

where $\|x\|_{\ell^1} := \sum_{j=1}^N |x_j|$ is the ℓ^1 -norm and $\lambda > 0$ is a rational parameter.

We provide a computability-theoretic explanation of the rampancy of hallucinations in the application of deep learning methods to inverse problems such as LASSO [1, 28]. Consider the solution mapping $\Xi : \mathbb{C}^{m \times N} \times \mathbb{C}^m \rightrightarrows \mathbb{C}^N$ which maps data (A, y) to the solution(s) $\Xi(A, y)$ of the LASSO problem (1.1). The double arrow indicates multi-valuedness of the mapping, since LASSO problems need not admit unique minimisers.

2.1 AI techniques replacing LASSO via training

A classical AI technique [19, 31] for computing a neural network to replace the LASSO method is as follows. Fix $k \in \mathbb{N}$ and a set of training pairs

$$\Omega \subseteq \{(A, \mathcal{S}) \mid A \in \mathbb{C}^{m \times N}, \mathcal{S} \subseteq \mathbb{C}^m \text{ has size } k\}.$$

Here, $\mathcal{S}$ is a set of vectors in $\mathbb{C}^m$, such that for each $y \in \mathcal{S}$, the pair (A, y) specifies a LASSO problem as in (1.1). Technically, Ω depends also on k ; with k fixed, we suppress this dependence. The goal in practice is to construct an algorithm that takes any such training pair $(A, \mathcal{S})$ in Ω and uses the data therein to produce a neural network approximating Ξ . We require our algorithms to be single-valued. Also, in reality, algorithms can only access *approximations* to the training pairs, via so-called “inexact inputs”, due to limitations on measurement capabilities and computer precision.

We assume these approximations can be made to an arbitrary level of accuracy via a recursive procedure, so that the inexact inputs are in fact represented by Turing machines. For a vector $x \in \mathbb{C}^N$ and a nonempty set $X \subseteq \mathbb{C}^N$, write $\operatorname{dist}(x, X) := \inf_{z \in X} \|x - z\|$ for the Euclidean point-to-set distance. We call a training pair $(A, \mathcal{S})$ *admissible* if it has at least one Turing-machine representation satisfying the conditions below, and henceforth restrict Ω to admissible training pairs. Such a representation supplies approximations to selected LASSO solutions in addition to approximations for A and the vectors in $\mathcal{S}$. Formally, let $\mathcal{TM}$ denote the set of Turing machines [46]; we use the classical Turing

model (often referred to as the “Markov model” [23, 40] of computation), where the inexact inputs are given by “computer codes” or, more precisely, Turing machines as follows. We say $\iota \in \text{TM}$ “represents” a training pair $(A, \mathcal{S}) \in \Omega$, where we enumerate $\mathcal{S} = \{y_1, \dots, y_k\}$, if $\iota(n)$ is (a code for) a $(2k+1)$ -tuple $\iota(n) = (A_n, y_{n,1}, \dots, y_{n,k}, x_{n,1}, \dots, x_{n,k})$, where

- $A_n \in \mathbb{Q}[i]^{m \times N}$ with $\|A_n - A\| < 2^{-n}$;
- $y_{n,j} \in \mathbb{Q}[i]^m$ with $\|y_{n,j} - y_j\| < 2^{-n}$, for $j = 1, \dots, k$;
- $x_{n,j} \in \mathbb{Q}[i]^N$ with $\text{dist}(x_{n,j}, \Xi(A_n, y_{n,j})) < 2^{-n}$, for $j = 1, \dots, k$.

Above, matrix differences are measured in the Euclidean operator norm and vector differences in the Euclidean norm. In other words, ι computes (approximations to) the entries of A , the vectors in $\mathcal{S}$, and solutions to the associated LASSO problems. The set of inputs to our desired algorithm is then

$$\Omega_\tau = \{\iota \in \text{TM} \mid \iota \text{ represents some } (A, \mathcal{S}) \in \Omega\}.$$

By a Gödel numbering, the elements of Ω_τ can themselves be read as inputs to other Turing machines. Observe that each $\iota \in \Omega_\tau$ represents a unique $(A, \mathcal{S}) \in \Omega$, whence we write $\iota = \iota_{A, \mathcal{S}}$. We seek a procedure that takes any training pair and returns a neural network approximation to the map Ξ . Let $\mathcal{N}_{m, N}$ denote the class of feedforward neural networks (NNs for short) from $\mathbb{C}^m$ to $\mathbb{C}^N$. For precise conventions on the representation and permitted architectures, see Section 1.B.1 of the supplementary information to [12]; see also [24] for background. For a prescribed accuracy parameter $K \in \mathbb{N}$, the question of interest is then:

Q: Can we construct an algorithm $\Gamma_K : \Omega_\tau \rightarrow \mathcal{N}_{m, N}$ such that, whenever $\iota \in \Omega_\tau$ represents $(A, \mathcal{S})$, the returned network satisfies

$$\sup_{y \in \mathcal{S}} \text{dist}(\Gamma_K(\iota)(y), \Xi(A, y)) \leq 10^{-K} \quad (1.2)$$

A training algorithm has K digits of accuracy on $(A, \mathcal{S})$ when its returned neural network satisfies the error criterion in (1.2). This criterion concerns approximation on the finite training set $\mathcal{S}$ only; behaviour away from $\mathcal{S}$ is a separate generalisation question.

See [12] for a precise definition of the term “algorithm” in this context; briefly, we mean a Turing machine (which, for each input, outputs

a rational encoding of a neural network with rational coefficients). Recall the universal approximation theorems: they are existence results for neural networks approximating a given (typically continuous) function to any desired accuracy (typically uniformly). The following result follows readily from the universal approximation theorems.

Theorem 1 (Existence of Neural Networks for LASSO [12]). *For any collection Ω as above, there exists a mapping*

$$\mathcal{K} : \Omega_\tau \rightarrow \mathcal{N}_{m,N}, \quad \mathcal{K}(\iota_{A,S}) = \varphi_{A,S}$$

with $\varphi_{A,S}(y) \in \Xi(A, y)$ for all $y \in \mathcal{S}$.

While Theorem 1 guarantees that for each input in Ω_τ there *exists* a neural network solving the associated LASSO problems on the corresponding training pair, it says nothing about *how* such a neural network can be found, if it *can* be found at all by a recursive procedure, and how this neural network behaves off of the training pair. Indeed, Theorem 1 stands in stark contrast to empirical results, which demonstrate the prevalence of instabilities, hallucinations, and other errors in trained neural networks for inverse problems [1, 9, 18, 28]. This tension between Theorem 1 and what is empirically observed is addressed by the following theorem. In this theorem, we say an algorithm uses M training data on input $\iota_{A,S}$ if the algorithm uses M many outputs of the Turing machine representing $\iota_{A,S}$.

Theorem 2 (Existence and Non-Existence of Accurate Neural Network Training Algorithms for LASSO [3, 12]). *Fix $\lambda \in (0, 1]$ as well as $m, N \in \mathbb{N}$ with $N > \max\{m, 1\}$. Let $K, L \in \mathbb{N}$ with $K > 2$. Then there exists an infinite class $\Omega = \Omega(K, L)$ of training pairs (A, S) as above, with the following properties. The class Ω is well-conditioned with respect to relevant condition numbers (briefly, conditioning measures sensitivity to minor deviations in the inputs to the matrices; see [11, 12, 39] for details), which are bounded by 1 independent of all parameters. Then the following all simultaneously hold:*

- (i) (**K digits of accuracy impossible**) *There does not exist any deterministic Turing machine that, given $\iota_{A,S} \in \Omega_\tau$, produces a NN with K digits of accuracy for each element of $\mathcal{S}$.*
- (ii) (**$K - 1$ digits of accuracy possible but requires arbitrarily many training data**) *There exists a deterministic Turing machine that, given $\iota_{A,S} \in \Omega_\tau$, produces a NN with $K - 1$ digits of accuracy for each element of $\mathcal{S}$. However, there is no uniform finite bound on the required training data: for every deterministic Turing machine that achieves*

$K - 1$ digits of accuracy on all inputs in Ω_τ and for every $M \in \mathbb{N}$, there exists an input $\iota_{A,S} \in \Omega_\tau$ on which the machine uses more than M training data.

- (iii) (**$K - 2$ digits of accuracy possible with L training data**) There exists a deterministic Turing machine that uses at most L training data for each $\iota_{A,S} \in \Omega_\tau$ and produces a NN which is accurate to $K - 2$ digits for each element of S .

Theorem 2 as stated in [12] is slightly weaker than the version stated above – which is in the Markov model. This slightly stronger version can be proved by adapting the arguments in [3]. Theorem 2 highlights key obstacles for training neural networks to solve inverse problems such as LASSO. Since we seek to train neural networks *in practice*, it is insufficient to establish merely their existence; computability thereof via an algorithm must be demonstrated. Indeed, given any desired number of digits of accuracy, there are LASSO problems for which no neural network can be trained to achieve this accuracy. Moreover, one can show that the above impossibility results cannot be remedied by using probabilistic methods. This implies that mainstream training methods, e.g. stochastic gradient descent, which are inherently probabilistic, also suffer from the ramifications of Theorem 2. In particular, just because stable and accurate neural networks can be proven to exist, they may not be computable by an algorithm. Hence, the neural networks computed in reality may (and in practice do) sometimes produce false information.

3 Smale’s 9th problem and extensions

The phenomena in Theorem 2 (and the proofs thereof) are situated in a well-established framework known as the Solvability Complexity Index (SCI) hierarchy [13, 21] and stem from Theorem 3 below. The SCI hierarchy is designed to analyse computational problems more general than decision problems. Moreover, this framework is not restricted to any particular model of computation; one may consider Turing machines or BSS machines [8], for example, or any other reasonable model of computation. In this way, the SCI hierarchy generalises the well-known arithmetical hierarchy [35, 44], which is based on computation in the Turing model. The program on the SCI hierarchy is also closely connected to S. Smale’s [41, 43] program on the foundations of computational mathematics and scientific computing. Smale’s program

motivated the early investigations of polynomial root-finding by C. McMullen [29, 30] and P. Doyle & C. McMullen [16]. Their work yielded pioneering classification results within the SCI hierarchy, predating its formal introduction. The SCI hierarchy permits the comparison of the difficulty of such computational problems by the number of limits it takes to solve them, and theorems then come in two forms: upper and lower bounds on computational difficulty. In particular, lower bounds in the SCI hierarchy expose limitations of AI methods to solve particular computational problems. As an illustrative example, we consider Smale's list of problems for mathematics in the 21st century [42]. Smale was interested in computational models which could handle approximate inputs and round-off computation, due to the reality of inaccurate data input in practice.

Linear programming (LP) concerns an optimisation problem of the form

$$\text{find } z \in \operatorname{argmin}_{x \in \mathbb{R}^N} \langle x, c \rangle \text{ subject to } Ax = y, \text{ and } x \geq 0,$$

where $A \in \mathbb{R}^{m \times N}$, $y \in \mathbb{R}^m$, $c \in \mathbb{R}^N$, and $x \geq 0$ means all entries of x are non-negative. Smale's 9th problem concerns polynomial time algorithms over the reals for LP, and an extended version of Smale's 9th problem concerns working with inexact input [42] (see Smale's discussion in [42] on inexact input). Theorem 3 below (and its proof, which can be found in [2]) is a prerequisite for Theorem 2 that highlights the essential role that logic and foundations of mathematics play in designing algorithms for solving mainstay optimisation problems such as LASSO and LP. In particular, any deep learning method – and indeed any AI method more generally – is subject to the limitations of Theorem 3. It follows that in order to develop trustworthy AI – in particular for scientific computing – a program on the foundations of AI via foundations of mathematics and logic is necessary. The framework of the SCI hierarchy is helpful for this task. In the following theorem, digits of accuracy are defined as in (1.2), where the distance in measuring the error is, again, the Euclidean point-to-set distance.

Theorem 3 (Extended Smale's 9th Problem [2]). *Consider the computational problem of computing a minimiser of LASSO or LP. For any integer $K > 2$ there exists a feasible set Ω of inputs (where inputs to algorithms are as in the LASSO set up above) such that:*

- (i) *There does not exist any algorithm which always halts and achieves K digits of accuracy. Allowing for probabilistic algorithms which, for*

each input, halt with probability one, no such algorithm can achieve K digits of accuracy with probability $p > 1/2$ on each input. Allowing for probabilistic algorithms that have a non-zero probability of not halting, there is such an algorithm achieving K digits of accuracy with probability $2/3$ on each input, and this is optimal; no such algorithm achieves K digits of accuracy with probability exceeding $2/3$.

- (ii) *There exists an algorithm (a Turing machine or a BSS machine, depending on the computational model) achieving $K - 1$ digits of accuracy, but it will take arbitrarily long to run; given any $T > 0$ and any such algorithm, the algorithm has runtime (see [2] for a precise definition) exceeding T on some input. Moreover, given any probabilistic algorithm and $p < 1/2$, there is an input for which, with probability exceeding p , at least one of the following occurs: either $K - 1$ digits of accuracy is not achieved or the runtime exceeds T .*
- (iii) *There exists an algorithm (Turing machine or a BSS machine) which achieves $K - 2$ digits of accuracy in at most polynomial time in the number of variables $m + mN$.*

4 Logic and the automated mathematician

We have seen that computability theory plays a crucial role in the hallucinatory phenomena exhibited by AI systems. We now turn our attention to the most prevalent AI system: the chatbot. Consider the prospect of a scientific computing chatbot, titled “SciCompBot”: a chatbot which accepts any question in scientific computing – for example optimisation – posed in natural or symbolic language, and returns an answer. A key question is therefore: *Could such a chatbot potentially replace computational mathematicians?*

Although modern chatbots such as ChatGPT and Claude already partially fulfill this very role, they also hallucinate profusely, delivering incorrect answers on simple scientific computing questions, even when told not to answer unless 100% confident [17, 20]. In other words, SciCompBot exists, but is not trustworthy. We consider the possibility of a trustworthy SciCompBot. If such a trustworthy SciCompBot can be developed, it could be capable of replacing – and arguably exceeding – computational mathematicians. In this way, the question of a trustworthy SciCompBot becomes one of considerable significance, and as we argue, yet another area of AI where logic is crucial.

4.1 Autoformalisation and the Navier-Stokes proof

An obvious task for SciCompBot is to translate mathematical proofs written in a natural language (NL) into a formal language such as Lean. Once this translation is done, the argument expressed, in for example Lean, can easily be mechanically verified. Such automated translation is referred to as autoformalisation, and has been a key component in OpenAI's recent announcement of blow-up of the Navier-Stokes equations. Indeed, their announcement contains a NL proof – aimed at human mathematicians – and a translation into Lean, which is then checked by the Lean compiler [4].

A pertinent question is therefore: *'does the correct compilation of the Lean proof imply correctness of the natural language proof?'* The answer to this question depends on whether the translation from the NL proof to Lean is semantically faithful, and herein lies the problem, this is far from straightforward. Indeed, for generations, logicians, mathematicians, linguists and philosophers have vigorously debated questions concerning natural and formal languages, semantics, and semantic preservation in translation [5–7, 25, 32, 33, 36, 37, 45].

It turns out that one can formulate the problem of providing semantically faithful translation as a computational problem that can therefore be classified in terms of the SCI. In particular, informally, we have the following [4]:

The problem of providing semantically faithful translation has $SCI = \infty$, whereas the Halting problem has $SCI = 1$.

This means, informally, that the problem of providing semantically faithful translation is harder than any problem which has $SCI < \infty$. Moreover, since the SCI hierarchy generalises the arithmetical hierarchy, this means that the problem of providing semantically faithful translation is harder than any problem in the arithmetical hierarchy.

There are several examples of recent cases, where mistranslations in AI autoformalisation are prevalent. The most notable example is probably OpenAI's announced proof of blow-up of solutions to the Navier-Stokes equations [4]. Another example is Meta's Atlas project, which 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 [38]. These claims were, however, swiftly refuted by the Lean community [4, 27], pointing out severe errors.

Remark 4 (Semantically faithful autoformalisation is not impossible).

Note that the above negative result does not mean that semantically faithful autoformalisation is impossible. Indeed, it is difficult, but semantically faithful autoformalisers can exist as long as they admit that there are cases they will give up on, and that they will reject certain texts and reply that faithful translations is impossible. In order to design such autoformalisers, we need to revisit the vast literature in logic and linguistics.

4.2 Creating a trustworthy scientific computing chatbot

We now show that tools from logic and foundations of mathematics have a key role to play in the development of a trustworthy SciCompBot. Let us restrict our attention to how a trustworthy SciCompBot would attempt to solve a particular problem. The computational spectral problem – to compute the spectrum $\text{sp}(T)$ of a bounded operator $T \in \mathcal{B}(\ell^2(\mathbb{N}))$ – and its variants have a rich classification theory in the SCI hierarchy [13, 21]. This problem arises widely in scientific computing [10, 13, 15, 21], in particular because of its connection to Schrödinger operators and quantum mechanics. If one restricts to normal banded operators, the spectral problem lies in the so-called Σ_1 class (see the SCI hierarchy as defined in [13, 21] for details), so that a trustworthy sequence of algorithms converging to the spectrum from below can be constructed. With this in mind, it becomes natural to ask whether we could design a sequence of algorithms $(\Gamma_n)_{n \in \mathbb{N}}$ such that both of the following hold:

- (1) For any banded normal operator T , we have convergence from below, i.e.

$$\Gamma_n(T) \rightarrow \text{sp}(T), \quad n \rightarrow +\infty, \quad \Gamma_n(T) \subseteq \text{sp}(T) + \overline{B_{2^{-n}}(0)};$$

- (2) If T is banded and non-normal, there is $N \in \mathbb{N}$ such that

$$\begin{aligned} \Gamma_n(T) &\subseteq \text{sp}(T) + \overline{B_{2^{-n}}(0)}, \quad n = 1, \dots, N, \\ \Gamma_n(T) &= \text{"I don't know"}, \quad n \geq N + 1. \end{aligned}$$

Here, $\overline{B_{2^{-n}}(0)}$ is the closed ball of radius 2^{-n} centred at the origin in $\mathbb{C}$, convergence is in the Hausdorff metric, and $\Gamma_n(T)$ is a finite subset of $\mathbb{C}$, or else denotes the output “I don't know”. The sum of sets above is the Minkowski sum. Such a sequence of algorithms would be trustworthy: it would converge from below on normal operators, and on non-normal

operators it would also remain correct until it eventually detects non-normality and henceforth returns “I don’t know”. Unfortunately, no such algorithm can exist.

Theorem 5 (Non-Existence of a Trustworthy Spectral Algorithm for Banded Operators with Properties (1) and (2) [20]). *For $k \in \mathbb{N} \cap [5, +\infty)$, let Ω consist of all banded bounded operators on $\ell^2(\mathbb{N})$ with no more than k subdiagonals and superdiagonals in total. Then, there does not exist a sequence of algorithms $(\Gamma_n)_{n \in \mathbb{N}}$ satisfying the above properties (1) and (2).*

Theorem 5 implies that building a trustworthy algorithm for the spectral problem for banded operators becomes rather delicate. How, then, would a trustworthy SciCompBot attempt to solve this spectral problem? The quintessential question is: *How can we make algorithms and AI that say “I don’t know” when they cannot solve the problem, and “I know” when they can solve the problem?*

Note that being able to distinguish problems one can solve from problems one cannot solve – and in the latter case say “I don’t know” – is a key feature of a competent human computational mathematician. One can argue that if an AI in the form of a SciCompBot were to replace computational mathematicians, such a SciCompBot must have this feature. The big problem is how should this be done?

As suggested above, a program on answering this question must have logic and foundations of computational mathematics at its heart. While the contours of this program have begun to be traced, this fundamental problem is far from having a solution.

References

- [1] V. Antun et al. “On instabilities of deep learning in image reconstruction and the potential costs of AI”. In: *Proceedings of the National Academy of Sciences* 117.48 (2020), pp. 30088–30095.
- [2] 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”. In: *arXiv:2110.15734* (2022).
- [3] A. Bastounis et al. “On the consistent reasoning paradox of intelligence and optimal trust in AI: The power of ‘I don’t know’”. In: *arXiv:2408.02357* (2024).
- [4] A. Bastounis, F. Circelli, and A. C. Hansen. “Navier-Stokes lost in translation: Why Lean verification of AI autoformalisation does not guarantee correct natural language proofs”. In: *arXiv:2610.08144* (2026).
- [5] J. van Benthem. “Natural Language and Logic of Agency”. In: *Journal of Logic, Language and Information* 23.3 (2014), pp. 367–382.
- [6] J. van Benthem and F. Liu. “New Logical Perspectives on Monotonicity”. In: *Monotonicity in Logic and Language*. Ed. by D. Deng et al. Berlin, Heidelberg: Springer Berlin Heidelberg, 2020, pp. 1–12.
- [7] F. Berto and L. Hornischer. “Cognitive synonymy: a dead parrot?”. In: *Philosophical Studies* 180.9 (2023), pp. 2727–2752.
- [8] 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”. In: *Bulletin of the American Mathematical Society* 21.1 (1989), pp. 1–46.
- [9] H. Boche, A. Fono, and G. Kutyniok. “Limitations of Deep Learning for Inverse Problems on Digital Hardware”. In: *IEEE Transactions on Information Theory* 69 (2023), pp. 7887–7908.
- [10] A. Böttcher. “Infinite matrices and projection methods”. In: *Lectures on operator theory and its applications (Waterloo, ON, 1994)*. Vol. 3. Fields Inst. Monogr. Amer. Math. Soc., Providence, RI, 1996, pp. 1–72.
- [11] P. Bürgisser and F. Cucker. *Condition: The Geometry of Numerical Algorithms*. Vol. 349. Grundlehren der mathematischen Wissenschaften. Berlin, Heidelberg: Springer, 2013.
- [12] 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”. In: *Proceedings of the National Academy of Sciences* 119.12 (2022), e2107151119.

- [13] M. Colbrook and A. C. Hansen. “The foundations of spectral computations via the Solvability Complexity Index hierarchy”. In: *Journal of the European Mathematical Society* 25.12 (2023), pp. 4639–4718.
- [14] E. Creamer. ‘Hallucinate’ chosen as Cambridge dictionary’s word of the year. *The Guardian*. Nov. 15, 2023.
- [15] T. Digernes, V. S. Varadarajan, and S. R. S. Varadhan. “Finite approximations to quantum systems”. In: *Rev. Math. Phys.* 6.4 (1994), pp. 621–648.
- [16] P. Doyle and C. McMullen. “Solving the quintic by iteration”. In: *Acta Mathematica* 163.3-4 (1989), pp. 151–180.
- [17] S. Farquhar et al. “Detecting hallucinations in large language models using semantic entropy”. In: *Nature* 630.8017 (2024), pp. 625–630.
- [18] N. M. Gottschling et al. “The Troublesome Kernel: On Hallucinations, No Free Lunches, and the Accuracy-Stability Tradeoff in Inverse Problems”. In: *SIAM Review* 67.1 (2025), pp. 73–104.
- [19] K. Gregor and Y. LeCun. “Learning Fast Approximations of Sparse Coding”. In: *Proceedings of the 27th International Conference on Machine Learning (ICML)*. Omnipress, 2010, pp. 399–406.
- [20] A. C. Hansen and F. Circelli. “Trustworthiness in AI: On SciCompBot – the Scientific Computing Chatbot”. In: *Philosophical Transactions of the Royal Society A* 384.2324 (2026), p. 20250116.
- [21] A. C. Hansen. “On the solvability complexity index, the n -pseudospectrum and approximations of spectra of operators”. In: *Journal of the American Mathematical Society* 24.1 (2011), pp. 81–124.
- [22] D. Heaven. “Why Deep-Learning AIs Are So Easy to Fool”. In: *Nature* 574.7777 (Oct. 2019), pp. 163–166.
- [23] P. Hertling. “A Banach-Mazur computable but not Markov computable function on the computable real numbers”. In: *Annals of Pure and Applied Logic* 132.2 (2005), pp. 227–246.
- [24] C. F. Higham and D. J. Higham. “Deep Learning: An Introduction for Applied Mathematicians”. In: *SIAM Review* 61.4 (2019), pp. 860–891.
- [25] T. F. Icard and L. S. Moss. “A Simple Logic of Concepts”. In: *Journal of Philosophical Logic* 52 (2023), pp. 705–730.
- [26] R. Kaur, D. Gabrijelčič, and T. Klobučar. “Artificial intelligence for cybersecurity: Literature review and future research directions”. In: *Information Fusion* 97 (2023), p. 101804.
- [27] Lean community. *Atlas (Meta)*. Zulip discussion in the Autoformalization channel. [Discussion thread](#). 2026.

- [28] Y. Li et al. “Evaluating Object Hallucination in Large Vision-Language Models”. In: *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*. Ed. by H. Bouamor, J. Pino, and K. Bali. Singapore: Association for Computational Linguistics, Dec. 2023, pp. 292–305.
- [29] C. McMullen. “Families of rational maps and iterative root-finding algorithms”. In: *Annals of Mathematics* 125.3 (1987), pp. 467–493.
- [30] C. McMullen. “Braiding of the attractor and the failure of iterative algorithms”. In: *Inventiones Mathematicae* 91.2 (1988), pp. 259–272.
- [31] V. Monga, Y. Li, and Y. C. Eldar. “Algorithm Unrolling: Interpretable, Efficient Deep Learning for Signal and Image Processing”. In: *IEEE Signal Processing Magazine* 38.2 (2021), pp. 18–44.
- [32] R. Montague. “Universal grammar”. In: *Theoria* 36.3 (1970), pp. 373–398.
- [33] Y. N. Moschovakis. “A Logical Calculus of Meaning and Synonymy”. In: *Linguistics and Philosophy* 29.1 (2006). doi:10.1007/s10988-005-6920-7, pp. 27–89.
- [34] R. Najjar. “Redefining Radiology: A Review of Artificial Intelligence Integration in Medical Imaging”. In: *Diagnostics* 13.17 (2023).
- [35] P. Odifreddi. *Classical Recursion Theory: The Theory of Functions and Sets of Natural Numbers*. ISSN. Elsevier Science, 1992.
- [36] W. V. O. Quine. “Meaning and Translation”. In: *On Translation*. Ed. by R. A. Brower. Cambridge, MA: Harvard University Press, 1959, pp. 148–172.
- [37] W. V. O. Quine. “On the Reasons for Indeterminacy of Translation”. In: *The Journal of Philosophy* 67.6 (1970), pp. 178–183.
- [38] A. Rammal et al. *Formalizing Mathematics at Scale*. arXiv:2605.29955. 2026.
- [39] J. Renegar. “Incorporating Condition Measures into the Complexity Theory of Linear Programming”. In: *SIAM Journal on Optimization* 5.3 (1995), pp. 506–524.
- [40] A. Salomaa et al. *Computation and Automata*. EBL-Schweitzer. Cambridge University Press, 1985.
- [41] S. Smale. “The fundamental theorem of algebra and complexity theory”. In: *Bulletin of the American Mathematical Society* 4.1 (1981), pp. 1–36.
- [42] S Smale. “Mathematical problems for the next century”. In: *Mathematics: Frontiers and Perspectives*. Ed. by V. Arnold et al. American Mathematical Society, 2000.
- [43] S. Smale. “Complexity theory and numerical analysis”. In: *Acta numerica*, 1997. Vol. 6. Acta Numer. Cambridge: Cambridge Univ. Press, 1997, pp. 523–551.

- [44] R. I. Soare. *Turing Computability: Theory and Applications*. 1st ed. Theory and Applications of Computability. Berlin, Heidelberg: Springer, 2016.
- [45] A. Tarski. "The Semantic Conception of Truth: and the Foundations of Semantics". In: *Philosophy and Phenomenological Research* 4.3 (1944), pp. 341–376.
- [46] A. M. Turing. "On Computable Numbers, with an Application to the Entscheidungsproblem". In: *Proceedings of the London Mathematical Society* S2-42.1 (1936), p. 230.

— About the author(s) —

Fabian Circelli is a PhD student in mathematics at the University of Cambridge, supervised by Professor Anders C. Hansen. His research focuses on the mathematical foundations of artificial intelligence and computation.

Anders C. Hansen is professor of mathematics at the University of Cambridge, where he leads the Applied Functional and Harmonic Analysis group, and at the University of Oslo. His interests include foundations of computations, recursion theory, AI and various forms of analysis.