European Journal for Philosophy of Science (2024) 14:4
https://doi.org/10.1007/s13194-024-00569-6

PAPER IN PHILOSOPHY OF TECHNOLOGY AND INFORMATION

Theorem proving in artificial neural networks: new frontiers in mathematical AI

Markus Pantsar[1,2]

Received: 16 February 2023 / Accepted: 10 January 2024 / Published online: 20 January 2024

Abstract

Computer assisted theorem proving is an increasingly important part of mathematical methodology, as well as a long-standing topic in artificial intelligence (AI) research. However, the current generation of theorem proving software have limited functioning in terms of providing new proofs. Importantly, they are not able to discriminate interesting theorems and proofs from trivial ones. In order for computers to develop further in theorem proving, there would need to be a radical change in how the software functions. Recently, machine learning results in solving mathematical tasks have shown early promise that deep artificial neural networks could learn symbolic mathematical processing. In this paper, I analyze the theoretical prospects of such neural networks in proving mathematical theorems. In particular, I focus on the question how such AI systems could be incorporated in practice to theorem proving and what consequences that could have. In the most optimistic scenario, this includes the possibility of autonomous automated theorem provers (AATP). Here I discuss whether such AI systems could, or should, become accepted as active agents in mathematical communities.

Keywords Automated theorem proving · Artificial intelligence · Machine learning · Artificial neural networks · Philosophy of mathematics

✉ Markus Pantsar
Markus.pantsar@gmail.com

[1] Human Technology Institute, Chair for Theory of Science and Technology, RWTH Aachen University, Theaterplatz 14, 52062 Aachen, Germany

[2] University of Helsinki, Helsinki, Finland

1 Introduction

Historically, proving theorems of mathematics was one of the central aims of artificial intelligence (AI) research (Newell et al., 1957). After some early successes, however, optimism about AI theorem proving faded. This does not mean that computer-assisted theorem proving or other computer applications have become less important for mathematical practice. In applied mathematics, in particular, the use of computer tools has become central to practice. This is an important topic, but in this paper, I will focus on the mathematical practice of proving theorems. Also in that regard, computers have had an important effect on mathematics as recent decades have seen an important rise in the status of computer assisted theorem proving. Many theorems, like the four-color theorem and the Kepler conjecture, have received proofs which would not have been possible without computers (Appel & Haken, 1976; Hales et al., 2017).[1] Nevertheless, the present generation of theorem proving computers is limited in its applications, and as a result the proofs are qualitatively different from proofs conducted by human mathematicians. Typically, present-day computer proofs apply brute computing force to exhaust a finite set of cases, one after another.

In addition to theorem proving, computer software are used to assist with logical deductions. In the field called *automated theorem proving* (ATP), such *proof assistant* software can be used for checking proofs, but also potentially for coming up with new proofs and new theorems. A typical automated theorem proving software functions by running specific algorithms to represent inferences in a system of logical calculus. Such rule-based software have many advantages. For one, we can generally trust them, because we can know the algorithm that it is running. However, the rule-based automated theorem provers have clear limitations. They run a mechanical procedure reliably, but their processing does not concern what the theorem is about, or even what the proof does to establish its correctness. In short, even though they are discussed as applications of artificial intelligence, they are not *intelligent* in any way.[2]

This is particularly important to remember when we consider the potential of computer software in proving new theorems, or providing new proofs for existing theorems, in an autonomous manner. An automated theorem prover can be fed a system of axioms that it then uses to prove and output theorems of the system. Usually in mathematics, the number of theorems is infinite, so the output needs to be limited to a finite subset. But even when limited to some finite subset of the theorems, the automated theorem prover would typically be indiscriminate in its proving capacity. It could list a million theorems as its output, but most likely only few of them would be in any way interesting to human mathematicians.

[1] One of the most important results in this regard was the proof of the so-called Robbins' conjecture, stating that all Robbins algebras (algebras consisting of a single binary and a single unary operator) are Boolean algebras, which was achieved by the theorem proving software *EQP*. For McCune's account of the proof, see: https://calculemus.org/MathUniversalis/4/6robbins.html. For a complete proof based on it, see (Mann, 2003).

[2] Here I purposely leave the word "intelligence" undefined and will not treat the topic in detail. An analysis of the concept of intelligence and its relevant sense to the present topic requires another paper. In this paper, I follow the custom of the field in talking about all theorem proving software as artificial intelligence applications, regardless of their putative intelligence.

One approach to make automated theorem provers more sophisticated is to program them to include criteria for interesting proofs and theorems. As we will see, some moderate progress has been made in this way. Yet the developments so far don't suggest that those types of software provide significant advances over previous programs. This matter would be potentially different, however, if instead of following specific rules, the computers *learned* mathematics. The theorem-proving potential of a particular type of artificial intelligence, i.e., a machine learning application run on *deep artificial neural networks* is the main topic of this paper. I will analyze how an artificial neural network could develop some way of processing mathematics that would enable it to distinguish between interesting and trivial proofs and theorems. As of now, symbolic mathematics is still a very difficult prospect for artificial neural networks. However, there are some early results that point out to the possibility that this could change in the future. In this paper, I will reflect on this possibility. In particular, I am interested in what kind of mathematical role this kind of AI could feasibly play, and how it would be received by human mathematicians and incorporated into the mathematical community.

In Sect. 2, I provide a short history of theorem proving in artificial intelligence, demonstrating how it has been an important issue in AI research ever since the establishment of the discipline. In Sect. 3, I will then present the state of the art in automated theorem proving, distinguishing between *interactive* and *autonomous* automated theorem proving. The current generation of automated theorem proving software is then analyzed in Sect. 4 in terms of their ability to distinguish between interesting and trivial proofs and theorems. In Sect. 5, the focus switches to machine learning and artificial neural networks, as I review some early results showing potential in the field. Then in Sect. 6, I provide a critical analysis of what artificial neural networks could and could not do in the field of theorem proving. In Sect. 7, I discuss the changes this would cause to mathematical practice in theorem proving and in Sect. 8 their epistemological importance. I argue that the changing epistemic role of computers in mathematics is best handled within a *community approach*, in which computer-assisted and computer-generated proofs are assessed by the mathematical community essentially similarly to the way humanly generated proofs are. Finally, in Sect. 9, I briefly discuss the questions of authorship and accountability that emerge from increasing use of AI in mathematics.

2 A very brief history of theorem-proving AI

Technically speaking, automated theorem proving is a subfield of the more general area in artificial intelligence research called *automated reasoning*. However, while the latter should in principle apply to different forms of reasoning, in practice automated reasoning has become largely identified with mechanized deductive inferences. Thus, the difference between the notions of automated reasoning and automated theorem proving has largely disappeared. Unless otherwise specified, both can be assumed to refer to mechanical, algorithmic computing procedures that represent inferences in formal systems of a logical calculus (Portoraro, 2021).

Therefore, the history of automated reasoning can be traced back to the formalization of logic. Particularly important in this development was the work of Frege, whose *Begriffsschrift* introduced a logical calculus of propositional and predicate logics (Frege, 1879). In his *Grundlagen der Arithmetik* (1884) and subsequent *Grundgesetze der Arithmetik* (1893), this approach was extended with the – ultimately unsuccessful – aim of deriving the laws of arithmetic from the basic laws of logic. The approach of using symbolic logic to derive mathematical theorems reached its early apex in the three-volume *Principia Mathematica* by Whitehead and Russell (1910–1914). Their goal was to show that all mathematical theorems can be derived from logical axioms by following rules of proof. If everything about mathematics could be thus *formalized*, it made sense to conjecture that mathematical reasoning could be consequently *automatized*. Indeed, there was early promise that this could be done when Presburger presented in 1929 an algorithm for deciding whether a sentence of an arithmetic consisting of natural numbers and addition is true (Presburger, 1929). However, this early promise was very quickly countered by Gödel's incompleteness theorems. While Presburger's arithmetic only had addition, Gödel showed that any formal system that can express standard Peano (1889) arithmetic (i.e., arithmetic with addition and multiplication) is in fact undecidable, i.e., it cannot prove all the truths of the system (Gödel, 1931).

In the study of mathematical logic, Gödel's result was momentous. Nevertheless, when technology developed sufficiently to make the mechanical application of algorithms reality, the incompleteness theorems did not discourage researchers of automated reasoning. One of the most important early developments in this regard was the 1954 programming of the Presburger algorithm into a vacuum tube computer by Davis. As reported by Davis (2001), Presburger's procedure was needlessly complex and the program didn't fare particularly well. It did manage, however, to prove that the sum of two even numbers is an even number (Davis, 1983). This may have been the first general mathematical theorem proved by a computer.

The work of Davis, however, was only one development in the rapid growth of AI research in the 1950s. Simultaneously with Davis, Newell, Simon and Shaw had been working on automated theorem proving and in 1956 they presented their computer program *Logic Theorist*, which proved theorems of *Principia Mathematica* of Whitehead and Russell (McCorduck & Cfe, 2004; Newell et al., 1957). *Logic Theorist is* often called the first artificial intelligence program (see, e.g., Crevier, 1993) and it was for its time quite impressive. It quickly proved 38 of the first 52 theorems of the Chap. 2 of *Principia Mathematica*.

Intriguingly for the present purposes, in one case the proof provided by the *Logic Theorist* was deemed more elegant than that provided originally by Whitehead and Russell (namely the theorem 2.85, see, (McCorduck & Cfe, 2004, p. 167). This may have well been the first case of an AI improving on the work of human mathematicians when it comes to theorem proving. It could have also become the first computer-assisted proof to be published in an academic journal. As recounted by Crevier (1993, p. 46), Newell and Simon submitted the proof to the *Journal of Symbolic Logic*. However, the paper was rejected on account of being in the outmoded system of *Principia Mathematica*, with apparently no significance given to the fact that a computer had come up with the proof.

3 Automated and interactive theorem provers

The great promise shown by early programs like *Logic Theorist* did not lead into the kind of revolution in mathematics that the first AI researchers may have envisioned. It certainly did not lead to the kind of *philosophical* revolution that one of its creators, Simon, later claimed:

> [W]e invented a computer program capable of thinking non-numerically, and thereby solved the venerable mind/body problem, explaining how a system composed of matter can have the properties of mind. (Simon, 1991, pp. 206–207)

This quotation may be too boastful for most people's tastes, and perhaps should not be taken at face value. But it also reveals an important belief of the early AI researchers. For them, there was no important difference between an AI showing human-like behavior and it thinking in a human-like fashion. This goes against a basic distinction standardly made in modern AI research, according to which we need to distinguish between intelligence as a property of *behavior* and as a property of *internal processes* (see, e.g., Russell & Norvig, 2020, Sect. 1.1). For Simon, because *Logic Theorist* showed intelligent behavior, it also had to have "properties of the mind".

In the modern context, this distinction is central. While the promise of the early theorem proving computers may not have (at least yet) been fully realized, in recent decades there have emerged many important theorem proving software, such as *Isabelle, Vampire, Prover9, Mizar, OTTER, Waldmeister, Lean* and *E*. In addition, software like *MATLAB* and *Mathematica* provide features that can be used for theorem proving purposes. All these software achieve far more than *Logic Theorist* ever could, but few would claim that they are in any way intelligent. Whatever the properties of the mind involved in theorem proving may be, the theorem proving software are not thought to mirror or instantiate them.

What the current generation of theorem proving software do in most common mathematical applications is roughly the following. The human user gives them a problem as the input, consisting of a set of axioms (first-order formulas) and a conjecture (a first-order formula). Then, standardly using first-order logic with equality, the theorem proving software checks whether the conjecture follows from the axioms (Voronkov, 2003, p. 1607).[3] Instead of a mere yes/no output from the theorem prover, it is desirable that the software produces a proof (in case of 'yes'), which should then be readable by humans (ibid.).

As mentioned in the introduction, such software for automated theorem proving are standardly called proof assistants. Sometimes they are also called *interactive theorem provers* (ITP). In interactive theorem proving, theorems are proved through a human-machine collaboration. A typical example of this is using the ITP for checking the validity of a formal proof, or a part of it. ITPs are used to different degree by many mathematicians and they are becoming increasingly important tools for the math-

[3] Instead of first-order formulas, the input can also consist of *clauses*, in which case the theorem prover checks whether the set of clauses is inconsistent (Voronkov, 2003, p. 1607).

ematical community (see, e.g., Barendregt & Wiedijk, 2005). The automated theorem prover *Mizar*, for example, is associated with a library (*The Mizar Mathematical Library*)[4] of formalized mathematical proofs that can be used by authors to check the validity of their proofs. These proofs currently formalize introductory mathematics, but new submissions are added constantly by the community to contribute more advanced results. This library could in the future provide an easy and reliable way to check the validity of proofs for state-of-the-art mathematical research.

This kind of interactive theorem proving is not the only way in which automated theorem provers can help mathematics progress. Another form of automated theorem proving would be for software to prove *new* theorems on their own. In such ATP applications, the idea is that the software proves theorems independently, after getting the initial input of a system of axioms. As a result, the ATP could both prove new theorems and provide new proofs to existing theorems. Let us call this *autonomous automated theorem proving* (AATP), to distinguish it from interactive theorem proving. In practice, the distinction between ITP and AATP is likely to be based on use, not necessarily on software. It is feasible that the same ATP software could be used for both purposes, even by the same mathematician.

4 Distinguishing between the interesting and the trivial

While the automated and interactive theorem provers have developed greatly in recent years, their importance for mathematical practice in the field of theorem proving should not be overestimated. Many mathematicians use such AI applications to varying degree in their work and in some tasks, like checking proofs, they can be an indisputably useful tool. In general, in the growing field of *experimental mathematics*, the use of computers for mathematical purposes has become increasingly important (see, e.g., McEvoy, 2013; B. van Kerkhove & van Bendegem, 2008). This approach can include testing conjectures, but also discovering new patterns and gaining new insights (Borwein & Bailey, 2008, pp. 3–4).

In this approach, the potential of AATP software is to be established. While they can provide proofs of new theorems, as well as provide new proofs of existing theorems, it remains to be seen how useful they can be in generating new theorems and proofs that mathematicians find *interesting*. What an ATP can do is take a system of axioms and derive proofs according to a system of logic. As the output, we could inquire whether a certain conjecture is a theorem of the axiomatic system, which is the ITP approach. Alternatively, we could simply have the ATP list a (finite) subset of theorems of the system, which is the AATP approach. What the AATP cannot currently do is evaluate the theorems it proves in terms of their mathematical importance. So far, to the best of my knowledge, there are no software that are somehow programmed to autonomously recognize interesting proofs, or interesting theorems.

[4] http://www.mizar.org/library.

That is still an exclusively human activity, and as such within the field of interactive theorem proving.[5]

This is not to say that automated theorem provers cannot discriminate between proofs based on human proof-theoretic criteria. The most obvious of these is the length of a proof. Veroff, for example, has presented a procedure for searching for the shortest proof with the theorem proving software *OTTER* (Veroff, 2001). Fitelson and Wos have also used *OTTER* to find shorter proofs for logical theorems (Fitelson & Wos, 2001). Kinyon has used the software *Prover9* for proof simplification, a procedure of shortening proof lengths (Kinyon, 2019). All these can be seen as efforts to find automated ways of establishing humanly appealing proofs. Yet these methods are very simple and take a limited approach even to the question of length of proofs.

To see this, we need to understand better how these software function. What the ATPs do is provide a list of inference steps and the justification for each step. Thus an ATP proof consists of two parts: a sequence of clauses consisting of atomic formulae and their negations, and the inferences used to derive the clause from its parenting clauses (Kinyon, 2019). The length of the proof refers then simply to the number of clauses in the sequence. Yet, as pointed out by Kinyon, the simplicity of the ATP proofs could also be measured by at least two other ways. First, instead of a sequence of clauses, a proof can be visualized as a directed graph. Simplicity of the proof could then refer to the complexity of such graphs. Second, in presenting the proof as a sequence of clauses, in addition to the number of lines (clauses) in the proof sequence, also the length of the clauses themselves adds to the complexity of the proof. This is measured in the simplest way simply by the number of symbols in each clause, called the *weight* of the clause (ibid.).

This gives us some idea how difficult it is to measure the simplicity of an ATP proof in an objective manner. So far, the approaches have focused on measuring the number of clauses in a proof sequence, but that is already a simplified procedure. However, even if we had a more inclusive measure, perhaps combining length with clause weights and graph complexity, how would we know to weigh the different notions in assessing the simplicity of a proof? One classic approach to find a way around such problems has been to invoke the notion of *Kolmogorov complexity* (Kolmogorov, 1963/1998). Kolmogorov complexity refers to the length of the shortest computer program which has an informative object, such as a string of symbols, as its output. To give a simple example, the string "bbbbbbbbbbbbbbbbbbbb" has a lower Kolmogorov complexity than the string "keehfydo38dkrislero29s". Both strings are 20 symbols long, but whereas the second string cannot (presumably) be described by a shorter string, the former can. The English description "20 times b", for example, is 10 symbols long (counting spaces). Thus, the former string has a lower Kolmogorov complexity than the latter.

ATP proofs can also be measured in terms of their Kolmogorov complexity, given that they are informative objects comparable to strings of symbols. This would have the advantage that instead of multiple measures, there would be a simple well-defined

[5] One exceptional approach was presented in (Lenat, 1976) in which an AI program was reported to use heuristics to evaluate the level of interest of theorems. These claims were heavily criticized (see, e.g., Ritchie & Hanna, 1984) and Lenat's approach has subsequently been largely ignored.

notion of complexity. Since it refers to the shortest computer program already in its definition, Kolmogorov complexity might initially appear to be suited as a general measure of simplicity of proofs. After all, mathematical proofs can be seen as instances of computer programs[6], and there is intuitive plausibility in the idea that shorter programs provide simpler proofs. However, Kolmogorov complexity is not without problems. While as a theoretical notion it may seem straight-forward and intuitive, it was proved already early on that Kolmogorov complexity is in fact incomputable, i.e., there is no general algorithm for determining the Kolmogorov complexity of a string of symbols (Vitanyi, 2020; see also Zvonkin & Levin, 1970). In addition, it has turned out that determining the Kolmogorov complexity of even short strings of symbols is an extremely difficult task (see, e.g., Soler-Toscano et al., 2014).

Through these kinds of considerations, it becomes clear that, at present, it is problematic to apply automated theorem provers even to assess the simplicity of a proof in a technical sense that the ATPs can process. This is to say nothing about the *cognitive* complexity of a proof. Proof lengths, clause weights and other such measures are related to the difficulty of the cognitive task of understanding a proof, but neither alone or together can they be equated with it. For this, we need a separate notion of cognitive complexity, one that takes into account particular aspects of human cognition, background knowledge and cultural context (Fabry & Pantsar, 2021; Pantsar, 2021a). This is the case if we focus on traditional measures of computational complexity or notions such as descriptive complexity (Pantsar, 2021b). As argued in those papers, computational complexity measures are rarely (if ever) directly applicable to studying complexity of cognitive tasks and processes.

Based on the above considerations, it is clear that the current generation of theorem proving AI applications lacks means of distinguishing between interesting and uninteresting proofs. Some minimal progress in terms of different understandings of simplicity has been made, but when it comes to having useful tools for discriminating proofs in terms of them being humanly interesting, the advances are negligible. In this respect, it is also important to note that simplicity is only one factor by which proofs are assessed by human mathematicians. Aside from simplicity, some notion of "insightfulness" is also likely to be present in assessing proofs (see, e.g., Macbeth, 2012; Weber, 2010). Another often mentioned property of mathematical proofs is their beauty. This topic has been discussed by philosophers in different ways (see, e.g., Johnson & Steinerberger, 2019; Rota, 1997). Recently, it has also been studied by neuroscientists and the experience of mathematical beauty appears to be a phenomenon associated with similar brain activity in the medial orbito-frontal cortex as other experiences of beauty (see, e.g., Zeki et al., 2014 for an experiment on the beauty of mathematical equations). Clearly such experiences related to mathematical proofs are not present in any way in the current generation of automated theorem proving software.

But even if we were able to assess proofs in any such way, it would not help us with the question of how the theorems themselves are assessed by human mathematicians. Why is some theorem considered to be important and another trivial? As with

[6] This is known as the Curry-Howard correspondence (see, e.g., Sørensen & Urzyczyn, 2006).

proofs, considerations of insightfulness and beauty can be relevant to this topic. But equally importantly, mathematical theorems get their importance as part of mathematical theories and their place within the mathematical community. Evaluating the importance of a mathematical theorem cannot be reduced to the mathematical properties of the particular theorem. Instead, the importance of a theorem is tightly connected to its place in the historical development of mathematics. The importance of Fermat's Last Theorem, for example, cannot be discussed without including its status in the mathematical community over centuries. Thus, all the considerations on mathematical importance – including insightfulness and beauty – need to be located in the context of wider mathematical practice.

In addition, one important factor in assessing the value of mathematical theorems is their applicability both in mathematics and wider in science (see, e.g., Lange, 2017). These, and many other questions concerning human mathematical practices are actively studied by philosophers of mathematics (for an introduction, see Mancosu, 2008). Here it is not possible to go further into details, but the problem should be clear by now. Human mathematicians associate proofs and theorems with a wide variety of valenced assessments. So far, automated theorem provers can only be included in such assessments in very rudimentary ways. For the big questions, i.e., why some theorem is considered to be important or interesting, or why some proof is considered to be more elegant than another, their current importance is negligible. When it comes to constructing artificial mathematical intelligence, the present automated theorem provers have little to contribute aside from their role as one tool at the disposal of the modern mathematician.

5 Artificial neural networks

The problems identified in the previous section relate to the present generation of automated theorem provers, which are rule-based systems. These systems function based on pre-set rules that the software follows to compute an output for a given input. This is what computer programs traditionally have been like: what they can do is constrained by the rules they were programmed to have. In such an approach to automated theorem proving, the question of distinguishing between different types of proofs and theorems is thus constrained by the kind of rules that are programmable. As we have seen, the length of an ATP proof can be one distinguishing factor in a rule-based system. However, factors like insightfulness, beauty and applicability would seem to be hopelessly too ambiguous to be included in rule-based systems. Certainly, there can be some programmable rules concerning such factors. For example, proofs by exhaustion (i.e., by the brute force method of verification case by case) are generally not considered to be elegant by human mathematicians. However, it does not seem feasible to capture notions like insightfulness or beauty by such rules. And if we don't have a good understanding of those notions in the first place, how can we realistically aim to program a computer to model them – not to say anything about the technical challenges involved in formalizing such notions into programmable forms even if we did have a good grasp of them.

In this respect, however, machine learning with (deep) artificial neural networks (ANN) can provide a different approach. In this type of artificial neural network (from here on just "neural network"), the computer learns to extract patterns from its input without being provided rules for it. ANN machine learning has made important advances in recent times and it has been applied highly successfully in fields like playing games (such as Go and chess), image recognition, natural language processing, and translation (for overview, see, e.g., Mitchell, 2019). Recently, there have also been advances in machine learning applications in mathematics.

Traditionally, artificial neural networks have been struggling with symbolic mathematics, but Lample and Charton (2019) have presented interesting data on a neural network that solved symbolic mathematical problems. The network was fed mathematical formulas (about 200 million of them) in a tree format and was given problems to solve (differentiation and differential equations; types of problems that do not have simple general-purpose solution algorithms). It performed well (and quickly, less than a second per problem) with 5000 test equations, giving right solutions to the vast majority of the problems. In integration tasks the success rate was almost 100%, in differential equations slightly less. Remarkably, for integration problems, it outperformed the standard commercial package *Mathematica* (Lample & Charton, 2019).

Such early results give hope that perhaps in the future, an ANN-based theorem prover can help human mathematicians in ways that automated theorem provers today cannot. Unlike a traditional rule-based theorem prover, such a neural network would have *learned* mathematics. If it were able to do that in a human-like fashion, it is possible that it could develop some sort of human-like ability to determine which theorems and proofs are interesting and which are not. For example, the AI could prove a million theorems in an axiomatic system and then rank them into categories in terms of their elegance, how interesting they are, and so on. If the AI is trained with the kind of theorems humans find interesting, it could develop an ability to recognize interesting theorems also among new theorems. The great advantage compared to the current generation of theorem provers would be that there would be no need to specify rules for what makes proofs and theorems interesting. Instead, these notions could remain implicit if the AI is able to detect a pattern in the training data. Presently this is of course in the realm of science fiction, but with the growth in AI development, such scenarios don't seem impossible.

Some reason for optimism is given already by the important progress that has been made in machine learning applications of interactive theorem provers. One of the key problems in this field is called *premise selection*. This refers to the problem of finding mathematical statements that are relevant for proving a particular conjecture (Wang et al., 2017). Progress has been reported in machine learning applications in the premise selection task using the *Mizar* library of formalized mathematics (Alemi et al., 2017; Wang et al., 2017). These approaches have led to progress in pre-selection of premises both for the first-order automated theorem prover *E* (Schulz, 2002) and higher-order logic theorem proving (Bansal et al., 2019). The general idea in these approaches is that machine learning limits the number of premises that are then used in a rule-based automated theorem proving software. This is far from developing autonomous automatic theorem provers, but it provides an important new research direction for interactive theorem proving. Under this approach, machine learning sys-

tems may not necessarily directly lead to proofs of theorems, but they could guide human mathematical intuitions and as such work as a new form of an interactive theorem prover (Davies et al., 2021). Indeed, machine learning applications have already been used to find counter-examples to open conjectures (Wagner, 2021).

It should be noted that all these developments are still in early stages and their importance should not be overstated. As pointed out by Davis (Davis, 2019), for example, there are several ways in which the ANN of Lample and Charton is not as impressive as first seems. First of all, it can handle only a limited subset of the problems that, e.g., *Mathematica* is able to solve. For problems solvable, or made considerably easier, by simplification, for example, rule-based systems like *Mathematica* would beat their ANN. Second, at times the output of the ANN is not even a well-formed formula, which is something that a rule-based system would not do. Third, and perhaps most importantly, it cannot be considered to be a stand-alone system. As Davis says, "the construction of [the ANN of Lample and Charton] is entirely dependent on the pre-existing symbolic processors developed over the last 50 years by experts in symbolic mathematics" (Davis, 2019, p. 6).

Nevertheless, it is clear that an ANN approach to automated theorem proving provides potential advantages that are beyond the capacity of the current generation of rule-based theorem provers. One important reason for this is that, as mentioned above, some of the notions that philosophers have suggested as criteria for interesting proofs and theorems, such as insightfulness and beauty, are too ambiguous to be captured by explicit rules. Machine learning systems, however, could detect patterns that correspond (at least partially) to the human interpretation of such notions. We should not expect this process to be accurate right from the beginning, but another strength of machine learning systems is that they are fast. With trial and error in creating and adjusting datasets for training the AI, progress can be made even if they do not function perfectly. Given the success of machine learning systems in many other tasks during recent years, I believe that we must start considering their potential in mathematics, as well as its significance philosophically.

6 The black box problem

We have seen that the current state of the art is far from providing feasible applications of autonomous automated theorem proving, let alone discriminating between interesting and trivial proofs and theorems. But in a philosophical discussion of automated theorem proving and artificial neural networks, we should not get stuck to the technical problems involved in the present generation of applications. However, there is one general difficulty concerning deep neural networks that we need to be concerned about, namely the "black box" problem (Russell & Norvig, 2020, Sect. 19.9.4). We do not have a clear idea what kind of explanations artificial neural network models provide, even when they are highly predictive (see, e.g., Kay, 2018). Deep neural networks learn, but often the only data we get of them is behavioral, i.e., concerning its output. How can we know that a theorem proving neural network has followed rules of proof correctly? Indeed, how could we make the neural network report on its reasoning? This question is closely related to the general problem in

the philosophy of science of the *epistemic opacity* involved in using computational methods in scientific explanations (Durán & Formanek, 2018; Humphreys, 2009).

To tackle the opacity involved in ANN mathematical processing, Lample and Charton (2019) suggest that we could trace the network's reasoning by making small adjustments to datasets and observe differences in behavior. However, in a scenario in which an artificial intelligence is trained to prove interesting new mathematical theorems, the effects could be exceedingly complex to determine. This general problem is well-known in AI research and the emerging area of *explainable AI* (XAI) aims to find solutions to the black box problem of machine learning (see, e.g., Doran et al., 2017; Holzinger, 2018; for an analysis of how this line of research connects to philosophy, see (Thompson, 2021). Many researchers, however, are skeptical of the possibility of explainable AI in deep neural networks due to the sheer complexity of the millions or even billions of terms in the equations involved in their processing. According to the skeptics, the best we can hope for is subjective interpretation of the neural network, not proper explanation (see, e.g., Landgrebe, 2022).

However, here theorem proving could potentially have different practical characteristics when compared to other types of problem solving, such as the cases reported by Lample and Charton. While with differential equations we might be content to simply get a solution, with new theorems mathematicians would expect something more. Instead of just presenting some formula as a theorem, we would ultimately expect the AI to provide some kind of humanly accessible proof of the theorem, as well. Therefore, the opacity of the artificial neural network would not prevent humans for evaluating the proof in a sufficient manner, even if the processing of the ANN would remain opaque. While the opacity of ANN would remain to be a problem, assessing the proofs would potentially give human mathematicians more information about the reliability of the ANN in the case of theorem proving.

It is important to recognize that AI theorem proving of the type we are discussing would not happen in a computer cocoon; rather, it would become part of the activity of the mathematical community. Thus AI-generated proofs could be inspected by human mathematicians, which could be a way to get around the black box problem. While the processing of the theorem proving software would remain a black box, the proof itself would be accessible to the scrutiny of mathematical peers – whether human or artificial. It is conceivable that in such a scenario the AI proof could be accepted along the same standards as human proofs. After all, in mathematics we are currently not concerned about the lack of knowledge about the human cognitive processes involved in proving theorems. What we are interested in are the theorems, their proofs, and partly informal expositions of them. Rather than transparency of the processing of the AI, presenting understandable proofs could be a more realistic aim for developing theorem proving artificial intelligence. This is the idea that I want to develop in the rest of this paper.

However, the above considerations prompt the question just how human-like the mathematical ability of the AI would be? Certainly, the learning process of the Lample and Charton design, at least, is very much un-human-like. No human mathematician will go through a training set of millions of formulas. Rather, the way humans learn is to use relatively few formulas – in addition to a lot of informal material – to capture the essential content involved in processes like integration and derivation.

Given the vast differences in the respective learning processes, it is reasonable to argue that the way deep neural networks learn things is at the very least a problematic fit with actual human learning.

This is important when we consider the problem of applying machine learning methods in proving theorems that mathematicians find interesting. While it is possible to generate training material of hundreds of millions of differential equations, the corpus of theorems that mathematicians find interesting is much smaller. It is thus questionable whether there could be a sufficiently large training set for an AI to learn to distinguish interesting theorems (or interesting proofs). Of course, *some* aspects of interesting theorems could be captured easily. For example, equations with identical left and right sides could feasibly be seen to be uninteresting. But overall, the phenomenon of interesting mathematics is likely to be so complex that the datasets would not be sufficient to detect the relevant patterns.[7]

Thus, the neural network AATP approach includes two difficult problems. First, the way the ANN learns is un-human-like, which may limit its applicability in recognizing humanly interesting mathematics. Second, the ANN could also be *un-ATP-like*, in the sense that we couldn't explain its processing. The great strength of the current generation of rule-based ATPs is that we can trust them to do their (limited) job correctly. With a machine learning ATP, we could no longer count on that.

7 Evolution of mathematical practice

How could ANN automated theorem provers be instilled in the mathematical practice of theorem proving? One possibility that we have already seen in the previous section is to include them in hybrid approaches that combine the use of machine learning with traditional rule-based systems. In this kind of approach, a neural network could come up with conjectures and relevant premises, and a rule-based system would prove it. This could potentially be a way around the black box problem, because we can trust the logical inferences conducted by the rule-based system.

Yet that kind of hybrid approach would not get us any closer to distinguishing between interesting and trivial theorems and proofs. The familiar question would remain in a new form, namely, whether an ANN could feasibly assess what is interesting and what is trivial as part of a hybrid system. In the previous section we saw that an ANN autonomous automated theorem prover would learn mathematics in a very un-human-like way. But perhaps it could follow its own rationality and establish criteria for what is interesting and what is trivial. These may or may not coincide with human assessments, but they might be a way forward in limiting the number of theorems and proofs that an AATP gives as an output. If humans ultimately evaluate the output, such limitations would be crucial.[8]

[7] It is likely that the training databases would need to be *generated*, which prompts the new problem of generating humanly interesting theorems that do not originate in human mathematics.

[8] One interesting suggestion I have heard is that interesting theorems are so rare among theorems that they could be akin to *anomalies* in data. Anomaly search in big data is an important research field and many techniques for it have been developed (for an overview, see Thudumu et al., 2020). Perhaps anomaly search among autonomous ANN-proved theorems could not by itself give us a set of interesting theorems,

But perhaps there could be an altogether different approach for a neural network to acquire human-like mathematical ability. While the ANN of Lample and Charton learned to solve problems by being fed exactly those types of problems as the input, could an ANN learn mathematics in a more bottom-up way, to mirror the way humans learn mathematics? Indeed, early advantages in such an approach have already been made. In experiments reported by (Stoianov & Zorzi, 2012; Testolin et al., 2020), an ANN learned numerosity discrimination from visual stimuli similarly to young children (Halberda & Feigenson, 2008; Piazza et al., 2010). This approach was extended also to counting (Di Nuovo & McClelland, 2019; Fang et al., 2018; Pantsar, 2023). So far, these abilities are very basic and even small integer addition is beyond the reach of the ANNs, but this kind of approach could lead to AIs learning mathematics in a more human-like manner. In that kind of learning process supervised by humans, the AI could also develop an ability to detect what kind of mathematics is interesting for humans.

But what would the status of such an AI, or any AI that has developed autonomous theorem proving abilities, be? If they were used as part of interactive theorem proving, it is feasible that they could enter theorem proving practice. Most likely this would not be without controversy, as we have seen in the history of introducing computer-assisted proofs into mathematics. From the four-color theorem (Appel & Haken, 1976) to the Kepler conjecture (Hales et al., 2017), there has been a significant change in how computer-assisted proofs in mathematics are seen. While the computer-assisted proof of the four-color theorem was criticized, among other things, for potential undetected errors (Tymoczko, 1979), the modern proofs like that of Kepler conjecture seem to be accepted more readily.

Indeed, this seems like a reasonable approach. While computer-assisted proofs may not be fully checkable, that is also the case with many human proofs. As proofs become longer and more complex, it becomes increasingly difficult for humans to check them in an error-free way. Thus, increasing the role of ATPs in mathematical practice, whether for proof-checking or proof-assisting, seems to be a justifiable future direction in mathematics. Ultimately, this could also include autonomous proving of theorems by ATPs, as well as autonomous writing of mathematical papers presenting the proofs.

8 The transforming epistemic role of computer tools in theorem proving

In the final scenario presented in the previous section, AI systems can autonomously prove theorems and write papers. Clearly this kind of epistemic role of computers would be vastly different from the present-day practice. But what exactly would that role be and how should we deal with it? It is good to remember that the initial reaction among philosophers to the introduction of computer-assisted proofs in mathematics

but in a hybrid approach it could significantly shrink the set of *potentially* interesting theorems. This set could then be assessed by human mathematicians.

was largely skeptical. The likes of Kripke (1980) and Tymoczko (1979) argued that the use of computers in theorem proving makes mathematics partly empirical.

This topic is important for the question of whether mathematical knowledge reached in this way can still be considered *a priori*. To appease such potential threats, Burge (1998) has convincingly argued that a proper kind of assimilation of computer use in theorem proving need not be essentially different from the kind of appeal to other mathematicians that we are happy to accept in knowledge-ascriptions. There are always *a posteriori* aspects related to mathematics – most trivially, we need to see symbols, etc. – but we may come to trust computers in similar ways that we trust human mathematicians. After all, no mathematician's knowledge of mathematics is based on understanding all proofs of known theorems. Sometimes our mathematical knowledge is simply based on trusting other people's competence. And as Burge rightfully asks, how is this essentially different from trusting the competence of a computer?[9]

The question Burge dealt with was whether computer-assisted mathematical proofs produce *a priori* knowledge. However, similar considerations are applicable also to the more fundamental question whether computer-assisted mathematical proofs produce knowledge in the first place, whether purely *a priori* or including empirical aspects. Nowadays, few mathematicians question computer-assisted proofs like that of the four-color theorem. By and large, mathematicians seem to accept that we can trust computers, even though we cannot completely discount the possibility of errors in their functioning. This seems sensible in light of Burge's analysis. For the most part, we accept that Andrew Wiles' proof of Fermat's Last Theorem, for example, is valid because we trust the relevant parts of the mathematical community.

However, the possibility of machine learning systems in theorem proving requires us to reassess the question whether computer-assisted proofs can produce knowledge. The kind of computer-assisted proofs that Burge discussed are firmly within the realm of rule-based systems, which were the only game in town back in 1998. Mathematics applying classical ATPs may be empirical only in the way that mathematics among humans is empirical (i.e., empirical aspects are present, but not in the *justification* of mathematical results). But could this be different for neural network AATPs? How do we come to trust such machine learning systems, and accept their results as mathematical knowledge?

It is instructive to approach also this question by comparing human mathematics with computer-assisted mathematics. Therefore, the first question to ask is how we come to accept human-proved theorems as mathematical knowledge. This is a complex social phenomenon where doubtlessly a variety of different factors can be identified. Reputation, for example, matters, as do extra-mathematical aspects like language skills. But hopefully we can assume that an important part of the acceptance procedure is the reviewing process in which the mathematical content is assessed by competent mathematicians. Ultimately, in this part of mathematical activity, review-

[9] A lot has been written on a related, but different topic of computer simulations (see, e.g., Barberousse & Vorms, 2014; Kaminski & Hubig, 2017; Symons & Alvarado, 2019). However, computer-assisted (or computer-generated) mathematical proofs and computer simulations of scientific phenomena require separate philosophical analyses. As noted by Symons and Alvarado (2019, p. 53), "The epistemic status of computer simulations is more akin to that of scientific instruments, than to mathematical proof."

ers are expected to focus primarily on the mathematical content. What they are not expected to focus on are the cognitive processes involved in coming up with the proof.

It is important to note that even though the process of accepting a mathematical theory into the canon of mathematical knowledge is thus focused primarily on the theorem and its proof, there are many important factors involved implicitly. For example, we typically trust that we share some sense of rationality among other humans. Among mathematicians, there is probably a heightened sense of that. For such reasons, it is not insignificant that we believe that a paper that we are reviewing was written by a *human*. We have come to accept – by and large – that the human practice of mathematics can include the application of computers also for providing parts of proofs that would not be otherwise possible. But this is still a different matter from accepting a proof that is entirely the product of a computer as mathematical knowledge.

However, I believe that this is mainly due to the unfamiliarity that this kind of technology evokes in us. Most importantly, of course, at present the technology does not exist. But given the problems that are associated with machine learning systems in other fields of AI, we can assume that similar considerations would arise also when (or if) AATPs were available. In this section, I want to prepare for that eventuality. More specifically, I want to discuss the scenario in which AATPs are not only available, but they are also *trusted* implicitly by humans. By this I don't mean that the AATP-generated proofs are accepted without scrutiny. On the contrary, I mean that they are trusted in the sense that human mathematicians are trusted, i.e., they can make errors but most of the time they function correctly.

Hence, I want to propose here a similar approach for the epistemic evaluation of machine learning AATPs as Burge (1998) proposed for rule-based computer-assisted proofs in mathematics. We should compare the scenario of accepting AATP-generated proofs to the mathematical practice of accepting human-generated proofs. Related approaches have been presented previously in the debate about computer-assisted proofs in mathematics. Detlefsen and Luker (1980), for example, argued against Tymoczko's (1979) skeptical view about computer-assisted proofs by remarking that humanly checked proofs are never absolutely certain, either. While this is often – including by Tymoczko, Detlefsen and Luker – understood as bringing a probabilistic, empirical element to mathematical proofs, I agree with McEvoy (2013) that this is mistaken. Rather than the proofs, it is our ability to *recognize* a genuine proof that is probabilistic.

Indeed, as argued by Hales (2008), there is good reason to think that ATP software (in his examples *HOL Light* and *Coq*) are actually particularly reliable ways of checking – and thus recognizing – proofs. They have a small "logical kernel" whose soundness can be reliably established. If anything, when it comes to classic ATP software, computer-assisted proofs can generally be established *more* certainly than "computer-free" proofs, simply due to the likelihood of human error as the proofs become longer. However, this clearly changes with AATPs based on machine learning. Such systems do not have a small logical kernel for their functioning, or if they do, we cannot determine what it is. Hence neural network AATPs face many similar questions about lack of certainty that computer-assisted proofs did in the 1970s and

80s. But just like those questions turn out, under proper analysis, to be about recognizing proofs, I contend that the related AATP-specific questions concern recognizing proofs. We should not disregard the possibility that an AATP-proven putative theorem turns out to be false, and in machine learning systems this possibility could indeed be greater than with classical ATPs. However, rather than being an argument against the use of machine learning systems for theorem proving, this is better understood as an invitation to develop accurate methods of checking proofs presented by such systems.

This question is tightly connected to the kind of output that neural network AATPs are designed to produce. At one extreme is a minimal output of simply stating that a particular string of symbols is a theorem of a particular mathematical system. At the other extreme is a full step-by-step proof of the theorem. In the former case, the checking of the proof would require constructing the proof, which would make the use of AATP minimally informative. In the latter case, the proof could be then checked by a classic rule-based ATP, in which case we could reach a maximally high standard of checking the validity of the proof.

In practice, it is likely that the output of the neural network would produce something between those two extremes. Since it would be trained with existing proofs of theorems, the output would most likely (in the realistic best-case scenario) bear a resemblance to human proofs. If this would indeed be the mode of the output, then the question of checking the validity of proofs becomes more intricate. The proofs would have gaps in logical steps, just like human proofs, which would pose challenges for the proof-checking process. However, in such a scenario, we would already have a system of mathematical practice ready. If the output of proofs is similar to humanly produced proofs, it seems reasonable that we would put the proofs under similar scrutiny as humanly produced proofs. This may involve using computers, or it can be purely human proof-checking. But the important point is that in such a scenario we would not discriminate between the proofs based on whether they are human-produced, hybrid-produced, or AATP-produced. They would all face the same level of scrutiny by the mathematical community. Let us call this the *community approach* to AI-generated mathematics.

The idea of the community approach is that AI systems, including possible AATPs, are assimilated to the mathematical community. This may happen in different ways. One way would be to make the application of an AI system explicit at all stages of practice. This would mean, for example, that proofs for which an AI application has been used are explicitly reported as such and the role of AI specified. However, it could also mean that an AATP could write a paper independently and it would be reviewed without the referees being aware that it is an AI-written paper. In this latter scenario, the community approach to AI-written papers would not differ in any way from that of human-written papers. Ultimately, in this scenario artificial agents would be accepted as part of the mathematical community.

9 Authorship and accountability

The above scenario of AATPs also presents potential problems, one of which concerns authorship. In the scenario, we should expect there to be papers with AIs both as co-authors and sole authors. But how could this work in practice? Could an AI actually be listed as an author? If not, why? Regulations for co-authorship including AI agents would need to be established but how could we make sure that AIs are credited according to those regulations? Indeed, a particularly problematic scenario is one in which an AATP creates a proof but a human mathematician takes credit for it. These types of problems have been widely reported in chess after chess-playing AI systems surpassed the level of human players. We should expect similar problems with regard to AI-generated mathematics. Mathematical practice is also about careers and the associated competition, which would become unfair if some mathematicians would be using (uncredited) AI applications for their mathematical work.

In addition to authorship, the question of *accountability* also needs to be discussed. How can we trust AI-generated mathematics and who is accountable if problems emerge? Every AI application, whether a commercial product or open source software, is produced by some group of people. In the scenario in which AATPs provide proofs of theorems, part of establishing trust in the AI system is establishing trust in its developers and users. As pointed out by, e.g., von Eschenbach (2021), AI is always situated within a socio-technical system that includes multiple groups of people, including AI designers but also administrators, legislators, marketers and many others, all the way to the end users of the software. In order to trust an AI application, von Eschenbach argues, we need to justify our trust in that socio-technical system. This kind of ethics-based approach may seem more relevant for AI applications in, for example, medicine, but it is also relevant for theorem proving AI. Mathematical theorems play an important role in technological and other scientific applications and establishing trust in them is crucial.

This question becomes particularly pertinent in a scenario in which an AATP-type AI is accepted as an independent mathematical agent, including a potential author of articles. In such cases it is important to establish where the authorship, and also hence the accountability, of the AATPs lies. Currently such questions may seem like science fiction, but they might well become important in the future. The future of mathematics is likely to become increasingly open to human-computer collaborations in which the human contribution gradually changes and, in terms of carrying out formal proof procedures, decreases. If the best way to achieve progress in mathematics is by increasing the amount of ATP use, this is likely to happen. In such a scenario, the human role in mathematical practice will change, and being able to apply ATPs may become a central skill. Among other things, this may force us to reconsider what it is to be a mathematician, and what higher mathematics education should be like. But it also raises important questions concerning the accountability involved in mathematics.

Acknowledgements This paper was written during my time as a Senior Research Fellow at the Käte Hamburger Kolleg "Cultures of Research", RWTH Aachen University. I would like to thank Stefan Buijsman and Daniel Wenz for helpful comments on an early version of the manuscript.

Authors Contribution As the sole author, I have contributed 100% of the paper.

Funding No further funding for this research was provided.
Open Access funding enabled and organized by Projekt DEAL.

Declarations

Financial or non-financial interests The author declares that there are no interests.

Ethical approval N/A, no experiments or data gathering took place.

Informed consent N/A.

Competing interests The author confirms that there are no interests to declare. No funding was received to conduct the research.

References

Alemi, A. A., Chollet, F., Een, N., Irving, G., Szegedy, C., & Urban, J. (2017). DeepMath—deep sequence models for premise selection. *arXiv:1606.04442 [Cs]*. http://arxiv.org/abs/1606.04442.

Appel, K., & Haken, W. (1976). Every planar map is four colorable. *Bulletin of the American Mathematical Society*, *82*(5), 711–712.

Bansal, K., Loos, S., Rabe, M., Szegedy, C., & Wilcox, S. J. (2019). HOList: An environment for machine learning of higher order logic theorem proving. *Thirty-Sixth International Conference on Machine Learning (ICML)*. https://arxiv.org/abs/1904.03241.

Barberousse, A., & Vorms, M. (2014). About the warrants of computer-based empirical knowledge. *Synthese*, *191*(15), 3595–3620.

Barendregt, H., & Wiedijk, F. (2005). The challenge of computer mathematics. *Philosophical Transactions of the Royal Society A: Mathematical Physical and Engineering Sciences*, *363*(1835), 2351–2375. https://doi.org/10.1098/rsta.2005.1650.

Borwein, J., & Bailey, D. (2008). *Mathematics by experiment: Plausible reasoning in the 21st century* (2nd edition). A K Peters/CRC Press.

Burge, T. (1998). Computer proof, apriori knowledge, and other minds: The sixth philosophical perspectives lecture. *Philosophical Perspectives*, *12*, 1–37.

Crevier, D. (1993). *AI: The tumultuous history of the search for artificial intelligence* (First Edition). Basic Books.

Davies, A., Veličković, P., Buesing, L., Blackwell, S., Zheng, D., Tomašev, N., Tanburn, R., Battaglia, P., Blundell, C., Juhász, A., Lackenby, M., Williamson, G., Hassabis, D., & Kohli, P. (2021). Advancing mathematics by guiding human intuition with AI. *Nature*, *600*(7887). https://doi.org/10.1038/s41586-021-04086-x. Article 7887.

Davis, M. (1983). *The prehistory and early history of automated deduction*. https://doi.org/10.1007/978-3-642-81952-0_1.

Davis, M. D. (2001). The early history of automated deduction. *Handbook of Automated Reasoning*. https://doi.org/10.1016/b978-044450813-3/50003-5.

Davis, E. (2019). The use of deep learning for symbolic integration: A review of (Lample and Charton, 2019). *arXiv:1912.05752 [Cs]*. http://arxiv.org/abs/1912.05752.

Detlefsen, M., & Luker, M. (1980). The four-color theorem and mathematical proof. *Journal of Philosophy*, *77*(12), 803–820. https://doi.org/10.2307/2025806.

Di Nuovo, A., & McClelland, J. L. (2019). Developing the knowledge of number digits in a child-like robot. *Nature Machine Intelligence*, *1*(12), 594–605.

Doran, D., Schulz, S., & Besold, T. R. (2017). What does explainable AI really mean? A new conceptualization of perspectives. *arXiv Preprint arXiv:1710.00794*.

Durán, J. M., & Formanek, N. (2018). Grounds for trust: Essential epistemic opacity and computational reliabilism. *Minds and Machines*, *28*(4), 645–666. https://doi.org/10.1007/s11023-018-9481-6.

Fabry, R. E., & Pantsar, M. (2021). A fresh look at research strategies in computational cognitive science: The case of enculturated mathematical problem solving. *Synthese*, *198*(4), 3221–3263. https://doi.org/10.1007/s11229-019-02276-9.

Fang, M., Zhou, Z., Chen, S., & McClelland, J. (2018). Can a recurrent neural network learn to count things? *CogSci*, 18.

Fitelson, B., & Wos, L. (2001). Finding missing proofs with automated reasoning. *Studia Logica: An International Journal for Symbolic Logic*, *68*(3), 329–356.

Frege, G. (1879). Begriffsschift. In J. Heijenoort (Ed.), *From Frege to Gödel: A source book in mathematical logic* (pp. 1–82). Harvard University Press.

Frege, G. (1884). *The foundations of arithmetic*. Basil Blackwell.

Frege, G. (1893). Grundgesetze der Arithmetik. In P. A. Ebert & Marcus (Trans.), *Rossberg as basic laws of arithmetic*. Pohle.

Gödel, K. (1931). On formally undecidable propositions. *Collected works: Vol. I* (pp. 145–195). Oxford University Press.

Halberda, J., & Feigenson, L. (2008). Developmental change in the acuity of the number sense: The approximate number System in 3-, 4-, 5-, and 6-year-olds and adults. *Developmental Psychology*, *44*(5), 1457–1465.

Hales, T. (2008). Formal proof. *Notices of the American Mathematical Society*, *55*.

Hales, T., Adams, M., Bauer, G., Dang, T. D., Harrison, J., Hoang, L. T., Kaliszyk, C., Magron, V., Mclaughlin, S., Nguyen, T. T., Nguyen, Q. T., Nipkow, T., Obua, S., Pleso, J., Rute, J., Solovyev, A., Ta, T. H. A., Tran, N. T., Trieu, T. D., & Zumkeller, R. (2017). A formal proof of the Kepler conjecture. *Forum of mathematics, Pi*, *5*. https://doi.org/10.1017/fmp.2017.1.

Holzinger, A. (2018). From machine learning to explainable AI. *2018 world symposium on digital intelligence for systems and machines (DISA)*, 55–66.

Humphreys, P. (2009). The philosophical novelty of computer simulation methods. *Synthese*, *169*(3), 615–626. https://doi.org/10.1007/s11229-008-9435-2.

Johnson, S. G. B., & Steinerberger, S. (2019). Intuitions about mathematical beauty: A case study in the aesthetic experience of ideas. *Cognition*, *189*, 242–259. https://doi.org/10.1016/j.cognition.2019.04.008.

Kaminski, A., & Hubig, C. (2017). Outlines of a pragmatic theory of Truth and Error in computer simulation. In M. Resch, A. Kaminski, & P. Gehring (Eds.), *The science and art of simulation I exploring—understanding—knowing* (pp. 121–136). Springer.

Kay, K. N. (2018). Principles for models of neural information processing. *Neuroimage*, *180*, 101–109. https://doi.org/10.1016/j.neuroimage.2017.08.016.

Kinyon, M. (2019). Proof simplification and automated theorem proving. *Philosophical Transactions of the Royal Society A: Mathematical Physical and Engineering Sciences*, *377*(2140), 20180034. https://doi.org/10.1098/rsta.2018.0034.

Kolmogorov, A. N. (1963). On tables of random numbers. *Theoretical Computer Science*, *207*(2), 387–395. https://doi.org/10.1016/S0304-3975(98)00075-9.

Kripke, S. A. (1980). *Naming and necessity*. Blackwell Publishers.

Lample, G., & Charton, F. (2019). Deep learning for symbolic mathematics. *arXiv Preprint arXiv:1912.01412*.

Landgrebe, J. (2022). Certifiable AI. *Applied Sciences*, *12*(3), 1050.

Lange, M. (2017). *Because without cause: Non-causal explanations in science and mathematics*. Oxford University Press.

Lenat, D. B. (1976). *An artificial intelligence approach to discovery in mathematics as heuristic search*. Stanford Univ Dept of Computer Science. https://apps.dtic.mil/sti/citations/ADA155378.

Macbeth, D. (2012). Proof and understanding in mathematical practice. *Philosophia Scientiæ. Travaux d'histoire et de Philosophie Des Sciences*, *16–1*, Article 16–1. https://doi.org/10.4000/philosophiascientiae.712.

Mancosu, P. (Ed.). (2008). *The philosophy of mathematical practice* (1st edition). Oxford University Press.

Mann, A. L. (2003). A complete proof of the Robbins conjecture. Http://Math.Colgate.Edu/~amann/MA/Robbins_complete.Pdf, 14.

McCorduck, P., & Cfe, C. (2004). *Machines who think: A personal inquiry into the history and prospects of artificial intelligence* (2nd edition). A K Peters/CRC Press.

McEvoy, M. (2013). Experimental mathematics, computers and the a priori. *Synthese*, *190*(3), 397–412. https://doi.org/10.1007/s11229-011-0035-1.

Mitchell, M. (2019). *Artificial intelligence: A guide for thinking humans*. Farrar, Straus and Giroux. Illustrated edition.

Newell, A., Shaw, J. C., & Simon, H. A. (1957). Empirical explorations of the logic theory machine: A case study in heuristic. *Papers presented at the February 26–28, 1957, Western joint computer conference: Techniques for reliability*, 218–230.

Pantsar, M. (2021a). Cognitive and computational complexity: Considerations from mathematical problem solving. *Erkenntnis*, *1–37*. https://doi.org/10.1007/s10670-019-00140-3.

Pantsar, M. (2021b). Descriptive complexity, computational tractability, and the logical and cognitive foundations of mathematics. *Minds and Machines*, *31*(1), 75–98. https://doi.org/10.1007/s11023-020-09545-4.

Pantsar, M. (2023). Developing artificial human-like arithmetical intelligence (and why). *Minds and Machines*, 1–18. https://doi.org/10.1007/s11023-023-09636-y.

Peano, G. (1889). The principles of arithmetic, presented by a new method. In H. Kennedy (Ed.), *Selected works of Giuseppe Peano* (pp. 101–134). University of Toronto Press.

Piazza, M., Facoetti, A., Trussardi, A. N., Berteletti, I., Conte, S., Lucangeli, D., Dehaene, S., & Zorzi, M. (2010). Developmental trajectory of number acuity reveals a severe impairment in developmental dyscalculia. *Cognition*, *116*(1), 33–41. https://doi.org/10.1016/j.cognition.2010.03.012.

Portoraro, F. (2021). Automated reasoning. In E. N. Zalta (Ed.), *The Stanford encyclopedia of philosophy* (Fall 2021). Metaphysics Research Lab, Stanford University. https://plato.stanford.edu/archives/fall2021/entries/reasoning-automated/.

Presburger, M. (1929). Über die Vollständigkeit eines gewissen Systems der Arithmetik ganzer Zahlen, in welchem die Addition als einzige Operation hervortritt. *Comptes Rendus Du I Congrès de Mathématiciens Des Pays Slaves, Warszawa*, 92–101.

Ritchie, G. D., & Hanna, F. K. (1984). A case study in AI methodology. *Artificial Intelligence*, *23*(3), 249–268. https://doi.org/10.1016/0004-3702(84)90015-8.

Rota, G. C. (1997). The phenomenology of mathematical beauty. *Synthese*, *111*(2), 171–182.

Russell, S., & Norvig, P. (2020). *Artificial intelligence: A modern approach* (4th edition). Pearson.

Schulz, S. (2002). E - a brainiac theorem prover. *AI Communications*, *15*(2,3), 111–126.

Simon, H. A. (1991). *Models of my life*. Basic Books.

Soler-Toscano, F., Zenil, H., Delahaye, J. P., & Gauvrit, N. (2014). Calculating Kolmogorov complexity from the output frequency distributions of small turing machines. *Plos One*, *9*(5), 96223.

Sørensen, M. H., & Urzyczyn, P. (2006). *Lectures on the Curry-Howard isomorphism* (Illustrated edition). Elsevier Science.

Stoianov, I., & Zorzi, M. (2012). Emergence of a 'visual number sense' in hierarchical generative models. *Nature Neuroscience*, *15*(2), 194–196.

Symons, J., & Alvarado, R. (2019). Epistemic entitlements and the practice of computer simulation. *Minds and Machines*, *29*(1), 37–60. https://doi.org/10.1007/s11023-018-9487-0.

Testolin, A., Zou, W. Y., & McClelland, J. L. (2020). Numerosity discrimination in deep neural networks: Initial competence, developmental refinement and experience statistics. *Developmental Science*, 23(5), e12940.

Thompson, J. A. F. (2021). *Forms of explanation and understanding for neuroscience and artificial intelligence*. PsyArXiv. https://doi.org/10.31234/osf.io/5g3pn.

Thudumu, S., Branch, P., Jin, J., Singh, J., & Jack (2020). A comprehensive survey of anomaly detection techniques for high dimensional big data. *Journal of Big Data*, *7*(1), 42. https://doi.org/10.1186/s40537-020-00320-x.

Tymoczko, T. (1979). The four-color problem and its philosophical significance. *The Journal of Philosophy*, *76*(2), 57–83. https://doi.org/10.2307/2025976.

van Kerkhove, B., & van Bendegem, J. P. (2008). Pi on earth, or mathematics in the real world. *Erkenntnis*, *68*(3), 421–435. https://doi.org/10.1007/s10670-008-9102-5.

Veroff, R. (2001). Finding shortest proofs: An application of linked inference rules. *Journal of Automated Reasoning*, *27*, 123–139.

Vitanyi, P. (2020). How incomputable is Kolmogorov complexity? *Entropy*, *22*(4), 408. https://doi.org/10.3390/e22040408.

von Eschenbach, W. J. (2021). Transparency and the black box problem: Why we do not trust AI. *Philosophy & Technology*, *34*(4), 1607–1622. https://doi.org/10.1007/s13347-021-00477-0.

Voronkov, A. (2003). Automated reasoning: Past story and new trends. *IJCAI*.

Wagner, A. Z. (2021). Constructions in combinatorics via neural networks. *arXiv:2104.14516 [Cs, Math]*. http://arxiv.org/abs/2104.14516.

Wang, M., Tang, Y., Wang, J., & Deng, J. (2017). Premise selection for theorem proving by deep graph embedding. *arXiv:1709.09994 [Cs]*. http://arxiv.org/abs/1709.09994.

Weber, K. (2010). Proofs that develop insight. *For the Learning of Mathematics*, *30*(1), 32–36.

Whitehead, A. N., & Russell, B. (1910). *Principia mathematica—volumes 1–3*. Cambridge University Press.

Zeki, S., Romaya, J., Benincasa, D., & Atiyah, M. (2014). The experience of mathematical beauty and its neural correlates. *Frontiers in Human Neuroscience*, *8*. https://www.frontiersin.org/article/https://doi.org/10.3389/fnhum.2014.00068.

Zvonkin, A. K., & Levin, L. A. (1970). The complexity of finite objects and the development of concepts of information and randomness by means of the theory of algorithms. *Russian Mathematical Surveys*, *25*(6), 83. https://doi.org/10.1070/RM1970v025n06ABEH001269.