Minds and Machines
https://doi.org/10.1007/s11023-020-09545-4

GENERAL ARTICLE

Descriptive Complexity, Computational Tractability, and the Logical and Cognitive Foundations of Mathematics

Markus Pantsar[1]

Received: 11 May 2020 / Accepted: 9 October 2020

Abstract

In computational complexity theory, decision problems are divided into complexity classes based on the amount of computational resources it takes for algorithms to solve them. In theoretical computer science, it is commonly accepted that only functions for solving problems in the complexity class P, solvable by a deterministic Turing machine in polynomial time, are considered to be tractable. In cognitive science and philosophy, this tractability result has been used to argue that only functions in P can feasibly work as computational models of human cognitive capacities. One interesting area of computational complexity theory is descriptive complexity, which connects the expressive strength of systems of logic with the computational complexity classes. In descriptive complexity theory, it is established that only first-order (classical) systems are connected to P, or one of its subclasses. Consequently, second-order systems of logic are considered to be computationally intractable, and may therefore seem to be unfit to model human cognitive capacities. This would be problematic when we think of the role of logic as the foundations of mathematics. In order to express many important mathematical concepts and systematically prove theorems involving them, we need to have a system of logic stronger than classical first-order logic. But if such a system is considered to be intractable, it means that the logical foundation of mathematics can be prohibitively complex for human cognition. In this paper I will argue, however, that this problem is the result of an unjustified direct use of computational complexity classes in cognitive modelling. Placing my account in the recent literature on the topic, I argue that the problem can be solved by considering computational complexity for humanly relevant problem solving algorithms and input sizes.

Keywords Complexity · Descriptive complexity · Computational complexity · Computationalmodelling · Mathematical cognition · Philosophy of mathematics · Philosophy ofcognitive science

✉ Markus Pantsar
markus.pantsar@gmail.com

[1] University of Helsinki, Unioninkatu 40A, P.O.Box 23, 00014 Helsinki, Finland

Published online: 09 November 2020

1 Introduction

In computational complexity theory, decision problems are divided into complexity classes based on the amount of computational resources it takes for algorithms to solve them. In theoretical computer science, it is commonly accepted that only functions for solving problems in the complexity class **P**, solvable by a deterministic Turing machine in polynomial time, are considered to be tractable, i.e., efficiently computable (Papadimitriou 1994; Arora and Barak 2007). In cognitive science and philosophy, this tractability result has been used (see, e.g., Frixione 2001; Gigerenzer et al. 2008) to argue that only functions in **P** can feasibly work as computational models of human cognitive capacities. This is called the *P-cognition thesis* in the literature (van Rooij 2008).

One interesting area of computational complexity theory is *descriptive* complexity, which connects the expressive strength of systems of logic with the computational complexity classes. One of the most important results of descriptive complexity is that while first-order systems (linearly ordered, with a least fixed-point operator) are connected with the complexity class **P**, richer logical systems yield complexity classes beyond what is considered tractable (Immerman 1999). In particular, given the generally accepted conjecture that the complexity class **NP**, the class of decision problems solvable by a *non*-deterministic Turing machine in polynomial time, is strictly greater than **P**, it follows that second-order systems of logic are considered to be computationally intractable. Hence, mirroring the above-mentioned reasoning in cognitive science, the proponents of the P-cognition thesis could argue that second-order logic is also unfit to model human cognitive capacities.

Such conclusion, while perhaps carrying some initial appeal, turns out to be unwarranted. I will argue that there are several ways in which it is problematic to draw direct connections between computational complexity, the complexity of logical languages, and computational models in cognitive science. However, the entire range of relevant cognitive phenomena is enormous and a thorough analysis is not possible in a single paper. For this reason, I focus on mathematics and mathematical problem solving to show that the computational complexity measures are too coarse to be used as general principles concerning the modelling of human cognitive capacities in these domains. The reason for this choice is two-fold. First, the differences between logical languages is a widely studied field in the foundations of mathematics and, as we will see, it provides us with philosophically interesting questions concerning the relation between logical and cognitive foundations of mathematics. Second, even though mathematical problem solving has its own characteristics, I believe that it can provide a platform for a more general treatment of how complexity should be applied in cognitive science. In the case of mathematical problem solving, the computational complexity measures are not sensitive to the actual problem solving algorithms used by human agents, nor are they sensitive to the computational characteristics of problems for bounded input sizes, only some of which are relevant for the modelling of human cognitive processes. On both accounts, there is no reason to believe that

mathematical problem solving is unique among cognitive phenomena. I will conclude that while computational complexity measures can work as rough guidelines in the computational modelling of cognitive capacities, strict limitations like focusing only on functions in **P** are unwarranted. This conclusion, I contend, can be generalised to other domains of cognitive phenomena.

Throughout this paper, I am not targeting a particular theory proposed in cognitive science or philosophy. I acknowledge that many researchers would not advocate a drastic use of computational complexity measures as limits to computational modelling in the cognitive sciences. Furthermore, to the best of my knowledge, descriptive complexity is not discussed in the current literature in cognitive science when it comes to mathematical cognition. Nevertheless, I believe that there are three reasons why the present topic is important to tackle. First, as we will see, computational complexity measures *are* an important part of the discussion on the theory of computational modelling, both among cognitive scientists and philosophers. While there have been important critical assessments of how computational complexity should be applied in cognitive science (e.g., van Rooij 2008; Isaac et al. 2014; Szymanik 2016; Szymanik and Verbrugge 2018; Van Rooij et al. 2019; Pantsar 2019b; Fabry & Pantsar 2019), limitations like the above P-cognition thesis are still too often misunderstood and given excessive importance. Second, these problems can extend to applications of descriptive complexity through the connections between computational complexity measures and the complexity of logical systems. Therefore, part of the purpose of this paper is to pre-empt potential confusions regarding descriptive complexity and the computational modelling of cognitive capacities, thus providing a feasible way in which computational complexity, and descriptive complexity in particular, can be applied in cognitive science and philosophy.

Third, I believe that mathematics and mathematical problem solving provide a clearly specified field of study for discussing computational complexity measures in the context of cognitive modelling, and this extends also to descriptive complexity. The mathematical consequences of adopting different logical systems as the foundation for mathematics is a much-researched topic. However, while perhaps implicitly present, these considerations have not been explicitly linked to the potential limits of human cognitive capacities. I will show that by connecting descriptive complexity measures to considerations on the modelling of cognitive processes, we can get a fruitful platform for discussing the relationships between logical and cognitive foundations of mathematics. In particular, I will show how drawing careless connections between the two can lead us astray.

I will begin in Sect. 2 by presenting computational complexity measures and how they have been used to argue for tractability principles when it comes to cognitive modelling. In Sect. 3, I will present the field of descriptive complexity and how it is connected to the computational complexity measures and other notions of complexity. Sect. 4 then presents a fundamental tension of logical systems as the foundations of mathematics: since many important results in mathematics require second-order logic, either we must commit to computationally intractable logic as the foundation of mathematics, or else we must use a system of logic that is too weak to express crucial mathematical concepts and/or prove theorems concerning them. In Sect. 5, I will then propose a solution to this tension by analysing the way the computational

complexity measures and descriptive complexity should be understood in the framework of human mathematical problem solving. I will then compare my account to two prominent alternatives in the literature, provided by van Rooij (2008) and Szymanik (2016). I will conclude that with well-considered specifications with regard to relevant input sizes and problem solving algorithms, tractability principles like the P-cognition thesis can be given their proper place as potentially useful guidelines, but not as anything resembling strict limits.

2 Computational Complexity and Tractable Cognition

In theoretical computer science, the complexity of decision problems is a fundamental question and it is standardly studied in the theoretical framework of Turing machines (Turing 1936).[1] Turing presented his model of a universal machine in order to study computation theoretically. Shortly put, a Turing machine functions by reading and writing symbols on a tape one at a time. The Turing machine is always in some inner state and based on the input symbol and the state, the machine has instructions to read and write symbols on the tape, move the tape, and change to a new inner state (or remain in the same state). The set of these instructions is called an *algorithm*. Since the Turing machine is a theoretical construct, no limits are made to the size of the tape or the number of operations that the machine can carry out.

According to the generally accepted *Church-Turing thesis*, if there is a mechanical procedure for solving a problem, then there is a Turing machine that can solve it (Turing 1936; Church 1936). This way, Turing machine has become the standard framework for studying computational complexity, but also for studying the problem-solving potential of algorithmic, mechanical procedures in general (see, e.g., Arora and Barak 2007). Consequently, in the study of complexity in theoretical computer science, researchers are not interested in the computing capacities of particular computers. Instead, they want to study the inherent complexities of different tasks free from the limitations of physical computers. Under this approach, the complexity of a mathematical problem can be characterised by the complexity of a minimally complex Turing machine (i.e., an algorithm run by a Turing machine that takes minimal resources) that solves the problem. Such an algorithm is called *optimal*.

Optimality of algorithms, however, is not a straight-forward matter. One important question concerns what the resource is that the algorithm should be optimal over. There are two common answers to that: time and space. Since the Turing machine is a theoretical construct, time and space are measured as functions of the size of the input, rather than any physical measure. But since there are no physical

[1] In this paper the focus is on decision problems, i.e., "yes/no" problems. There are also many other types of problems studied in theoretical computer science, including counting, search, function and optimisation problems (Goldreich 2008). I focus on decision problems because it is the most researched field in the present context. It also needs to be noted that many problems can be framed in different ways, so these distinctions can be somewhat arbitrary.

limits to the Turing machine, there are also no limits to the size of the input. That is one reason why the complexity measures based on the Turing machines are *asymptotic*: they characterise the complexity of algorithms as the input sizes grow without limit.[2]

The great advantage of the asymptotic complexity measures is that they can be used to divide problems into *complexity classes.* The plethora of different complexity classes and the relationships between them is an active topic of research and it is not possible to go extensively into the details here (for more, see Papadimitriou 1994; Arora and Barak 2007).[3] But taking time as the relevant measure, we can use two complexity classes to show how complexity classes work and what kind of relevance they can have for philosophy. One of the most important complexity classes is called **P** (or **PTIME)** and it is defined as the class of decision problems that can be solved by a deterministic Turing machine in polynomial time. An algorithm (i.e., a Turing machine) is said to run for polynomial time if its running time has an upper bound of a polynomial function of the size of the input for the algorithm. This means that if the size of the input is n, the running time has an upper bound of some function n^k for some constant k. Another complexity class relevant here is called **EXP** (or **EXPTIME**) and it is the class of decision problems that are solvable by a deterministic Turing machine in *exponential* time. An algorithm runs for exponential time if its running time has a lower bound of some exponential function of the size of the input, i.e., for input size of n, the running time has a lower bound of some function $2^{p(n)}$ where $p(n)$ is some polynomial function of n.

Here I have wanted to present **P** and **EXP** for two reasons. First, following the so-called Cobham's thesis (or Cobham-Edmonds thesis), **P** is standardly accepted as the class of problems that can be feasibly solved by a computer (Cobham 1964; Edmonds 1965). Second, it has been proven that **EXP** is strictly greater than **P**. Algorithms for solving problems in **P** are called *efficient* or *tractable* (Garey and Johnson 1979). Algorithms for solving problems in **EXP** (that are not in **P**), on the other hand, are considered to be *inefficient* or *intractable* (ibid.). A simple example of an efficient algorithm is the standard schoolbook algorithm for multiplying integers, which takes roughly n^2 steps of computation for two n-digit integers. The standard algorithm for factoring integers into primes, on the other hand, is a good example of an inefficient algorithm. An n-digit integer takes about $2^{n^{1/3}}$computational steps (Pomerance 1996).[4]

[2] It should be noted that optimal algorithms are not unique. In theoretical computer science, an algorithm is called asymptotically optimal if it never performs more than a constant factor worse than the best possible algorithm. There can thus be many (even an infinite number of) optimal algorithms. It should also be added that although the method of characterising complexity of a problem in terms of an optimal algorithm for solving it is commonplace, we know from Blum's speedup theorem (Blum 1967) that it is not possible in all cases to define the computational complexity of functions in terms of optimal algorithms for solving them.

[3] For the state-of-the-art, see Scott Aaronson's highly informative website "Complexity Zoo" (https://complexityzoo.uwaterloo.ca/Complexity_Zoo).

[4] It should already be noted here that for inputs of fixed sizes, a problem in **EXP** can be less complex to solve than a problem in **P**. As mentioned above, the complexity classes are asymptotic and as such characterize the complexity of an algorithm as the input sizes grow without limit. More on this topic in Sect. 5, where it will play an important role.

While it is proven that **EXP** is strictly greater than **P**, perhaps the most important inclusion relation between the complexity classes is still a matter of conjecture. The complexity class **NP** is defined as the class of decision problems that can be solved by a *non-deterministic* Turing machine in polynomial time.[5] It is easy to see that **P** is a subclass of **NP**, but it has proven to be difficult to show whether it is a *proper* subclass of it, i.e., whether $\mathbf{P} \neq \mathbf{NP}$. Although this is perhaps the most famous unsolved problem in theoretical computer science, it is a generally accepted conjecture that the class **NP** is strictly greater than **P**. This is important for the present context since because the conjecture $\mathbf{P} \neq \mathbf{NP}$ is generally accepted, the complexity class **NP** is *de facto* the lowest complexity class of problems which are thought to be computationally intractable.

For computer science, the distinction between tractable and intractable algorithms is central, but what is its relevance for philosophy and the cognitive sciences? It turns out that this distinction can be potentially useful for characterizing *cognitive* complexity, i.e., the complexity of human cognitive tasks and processes. In computational modelling in the cognitive sciences, the focus is often on what Marr (1982) calls the *computational level of explanation*. This means that rather than focusing on the actual algorithms used in performing a cognitive task (the *algorithmic level* in Marr's terminology), or the neuronal activity (Marr's *implementational level*), the focus is on identifying a mathematical function that can model a cognitive process (Marr 1982; Pantsar 2019b). This is based on the understanding of cognitive tasks as employing cognitive capacities to transfer input states (e.g., perceptions) into output states (e.g., decisions), thus yielding a function that models the cognitive task as input-output mappings (Cummins 2000). In the computational-level approach, it is then possible to study these functions in terms of their computational complexity.[6] This computational-level framework has been highly influential for studying cognitive capacities (see, e.g., Newell 1982; Pylyshyn 1984; Horgan and Tienson 1996). As explained by Frixione:

> The aim of a computational theory is to single out a function that models the cognitive phenomenon to be studied. Within the framework of a computational approach, such a function must be effectively computable. However, at the level of the computational theory, no assumption is made about the nature of the algorithms and their implementation. (Frixione 2001, 381)

[5] The difference being that whereas a deterministic Turing machine is programmed to have one action for each situation, a non-deterministic Turing machine can pick the action from several options.

[6] A fundamental question in all this is whether human cognition should generally be considered to be computational, and in particular whether there are forms of mathematical cognition that cannot be captured by algorithmic procedures. Such a position was suggested by Turing (1948) and was argued for explicitly by Lucas (1961) and Penrose (1989, 1994). It is not possible to go into the details here (for more, see, e.g., Piccinini (2003)), but I work under the assumption that there are no theoretical limitations in the algorithmic approach to modelling mathematical cognition. In addition, I assume that cognitive processes can generally be captured by computational models (without taking a stand in the question whether cognitive processes *are* computation in some more substantial sense). In particular, against the arguments of Lucas and Penrose, I do not see any reason why the human mathematical ability could rise above all algorithmic or mechanical procedures, i.e., Turing machines. See Pantsar (2009) for more.

This way, many researchers accept that computationally intractable functions cannot accurately model human cognitive tasks. This has become known as the *tractable cognition thesis* in the literature (van Rooij 2008; Isaac et al. 2014). According to the tractable cognition thesis, when we are looking for functions that can potentially model cognitive capacities, we should limit ourselves to those functions that can be computed by tractable algorithms. Standardly, based on Cobham's thesis, this has been understood as the P-cognition thesis, stating that we should limit our considerations to functions associated with problems in the complexity class **P** (Arora and Barak 2007; van Rooij 2008; Van Rooij et al. 2019). Many cognitive scientists have accepted the P-cognition thesis as a fundamental rule of computational-level explanations. For example, according to Gigerenzer and colleagues:

> The computations postulated by a model of cognition need to be tractable in the real world in which people live, not only in the small world of an experiment with only a few cues. This eliminates NP-hard models that lead to computational explosion (...) (Gigerenzer et al. 2008, 236; quoted in Van Rooij et al. 2019, 17).

Thus the approach of Gigerenzer and others draws a direct link between computational complexity and cognitive complexity. In the case of mathematical problem solving, this implies that mathematical cognitive processes should be computationally modelled by functions whose values can be computed by algorithms for solving decision problems in the complexity class **P**.[7]

With Regina E. Fabry, we have argued that such a direct connection between computational and cognitive complexity is potentially problematic since the computational complexity approach does not transfer in a straight-forward manner into studying the complexity of human cognitive processes (Pantsar 2019b; Fabry & Pantsar 2019). One of the reasons for this is that human mathematical problem solving often includes visual and heuristic reasoning (e.g., the use of diagrams and spatial manipulation of symbols (Fabry & Pantsar 2019)) that is not usually included in the algorithmic approach to problem solving used in theoretical computer science (Pantsar 2019b). Thus it is important to realize that human mathematical problem solving has its own particular characteristics that do not always correspond to the computational complexity approach. In this paper, I want to extend the critical evaluation of applying results from computational complexity theory to cognitive modelling to one important subfield of computational complexity, which has not been extensively studied in the philosophical or cognitive scientific literature so far. This subfield is the complexity of systems of *logic*.

[7] As has probably become clear by now, in this paper I talk about cognitive processes in a general way rather than the cognitive processes of particular individuals. This follows the standard usage in the relevant literature and corresponds roughly to Chomsky's (2015/1965) distinction between *competence* and *performance*, which was also used by Marr (1982). More specifically, here I am interested in *enculturated* competence in mathematical problem solving, i.e., the culturally-shaped competence we have in mathematics. See Fabry & Pantsar (2019) for more.

3 Descriptive Complexity

The computational complexity measures such as **P**, **NP** and **EXP** are standardly used for classifying decision problems, but they have also turned out to have important connections with languages of logic. This field of study is called *descriptive complexity* (Immerman 1999). Typically in computational complexity theory, we ask about the complexity of the task of checking if a certain input has a certain property. In descriptive complexity, we ask how complex it is to express that property in a formal language (Immerman 1995). This way, rather than measuring complexity in terms of time or space requirements for solving decision problems, as in standard computational complexity theory, descriptive complexity measures complexity in terms of the strength of logical systems. As Immerman argues, time and space are natural complexity measures from an engineering standpoint since they quantify physical resources required of completing a computational task, but neither appears to capture the inherent *mathematical* complexity of computational problems (Immerman 1999, 1). Thus, when we consider the cognitive complexity of mathematical problem solving processes, descriptive complexity initially appears to have great relevance. The logical systems we use have different strengths in expressing and proving mathematical statements, and it is *a priori* feasible that processing different logical languages also differs in terms of the complexity of the associated cognitive processes.[8]

What makes descriptive complexity particularly interesting to the present discussion is the way it has proven to be directly connected to results concerning computational complexity classes. The research field of descriptive complexity can be traced back to Fagin's (1974) result that the complexity class **NP** is equal to the class of problems describable in the existential fragment of second-order logic (also known as Σ^1_1-logic).[9] Many other such connections have since been proven. Full second-order logic, for example, has been shown to yield the complexity class **PH**, which is the union of all complexity classes in the polynomial hierarchy (Stockmeyer 1977). **PH** of course contains **NP** but it is also thought to contain many stronger complexity classes, making full second-order logic more complex than its existential fragment.[10]

[8] It is important to note that the strength of a logical system required to express a mathematical statement may differ from that of a system required to *prove* that statement (generally or in particular cases). Here I will focus mainly on the strength required for expressing mathematical concepts, under the assumption that proving statements about those concepts is generally not possible with a weaker logical system. However, this can be possible for particular instances.

[9] In existential second-order logic, one can have universal quantification only over first-order objects, but not on second-order objects such as relations and sets. Existential quantification is allowed over both first- and second-order objects.

[10] Although all this is a matter of conjecture: if it turned out that **P = NP**, it would also be the case that **P = PH** (Hemaspaandra 2018). In any case, as Immerman (1999) points out, **PH** is a rather strange complexity class because it implies having an exponential number of Turing machines but limiting them to run for a constant time. The complexity class **PSPACE** is more much intuitive, as it allows for the time to grow polynomially. **PSPACE** is the class of all decision problems solvable in polynomial space and it is generally thought to be strictly greater than **PH** (Aaronson 2009). This would make it even stronger than full second-order logic.

Here it is not possible to go into all the intricacies of the connections between particular cases, but it is essential to see the general connection between the expressive strength of logical systems and the computational complexity of classes of decision problems. It is particularly important to note that there is a crucial difference between first-order logical systems and second-order systems in terms of their complexity. While existential second-order logic yields the complexity class **NP**, classical first-order logical systems give us the complexity class **P**, or one of its subclasses.[11] Therefore, only systems of first-order (classical) logic are connected with complexity classes that are considered to be computationally tractable. Even if it turned out to be the case that **P = NP**, this would still be a problem. It is known that second-order logic with a least fixed point operator yields the complexity class **EXP**, which we know to be strictly greater than **P** (Immerman 1999).[12]

This connection between computational complexity classes and the descriptive complexity of systems of logic has dual importance for the present purposes. First, it tells us that - according to the generally accepted view - systems of second-order logic are considered to be computationally intractable. But this is not limited to considerations on computational complexity. In the framework of computational modelling of cognitive capacities, this has important consequences also when it comes to cognitive complexity. If we accept the P-cognition thesis, we also need to accept that second-order logic, even the existential fragment of it, can express functions too complex to feasibly model human cognitive capacities, since many functions expressed in second-order logic are not in the complexity class **P**. In other words, by using second-order logic to describe functions supposedly modelling human cognitive capacities, we run the risk of including intractable functions, which goes against the P-cognition thesis.

Second, the connection between second-order logical systems and intractable computational complexity classes tells us that many important mathematical results can only be expressed with systems of logic that are computationally intractable. For example, the (upward) Löwenheim-Skolem theorem famously shows that first-order theories cannot distinguish between the cardinality of infinite models. If a first-order theory has an infinite model of one transfinite cardinality, it has a model of every transfinite cardinality (Hodges 1993). This would be a huge problem in mathematics because, among other problematic aspects, it means that in first-order theories it is not possible to distinguish between the cardinality of natural numbers and the cardinality of real numbers. There are numerous known differences between first-order and second-order logical theories and, by and large, the latter are seen as an important part of the toolbox of logicians and mathematicians (Väänänen 2019). Thus if,

[11] Standard first-order logic is a very weak system that corresponds to the complexity class $\mathbf{AC^0}$, a system of circuit complexity which is known to be strictly smaller than **P**. However, $\mathbf{AC^0}$ does not contain even integer multiplication (Vollmer 1999). Linearly ordered systems of first-order logic with a fixed-point operator are the strongest systems known to be included in **P**.

[12] Since the least fixed point operator shows up for the second time, it should probably be noted that it is important in complexity considerations because of its strength in allowing recursive definitions.

based on the P-cognition thesis, we limit the complexity of acceptable logical languages, it would go against the prevalent use of logical systems in mathematics.

However, before we go deeper into the potential problems in applying descriptive complexity measures for the foundations of mathematics, we need to be more detailed about descriptive complexity and its applications. In particular, we should first consider what kind of systems and problems descriptive complexity concerns. As Grohe (1999) puts it, one main theme of descriptive complexity is to study *model-checking* problems, i.e., problems of the type:

> Given a finite structure $\boldsymbol{A}$ and a sentence ϕ of some logic $\mathscr{L}$, decide whether $\boldsymbol{A}$ satisfies ϕ. (Grohe 1999, 14).

It is important to note that model-checking complexity is just one notion of complexity used in the literature, and there are substantial differences between different notions of complexity. One important notion of complexity, for example, is *expression complexity* which measures the complexity of a formula in terms of the length of its expression in different models (Vardi 1982). As Szymanik (2016, 104) remarks, expression complexity and model-checking complexity have a potentially important difference: the existential fragment of second-order logic (Σ^1_1-logic) is of the expression complexity **NEXPTIME** while its model-checking complexity is **NP**-complete. This prompts the question why we should focus on model-checking complexity when another notion of complexity like expression complexity (or perhaps a combination of two or more notions of complexity) could be more relevant? This is particularly important when we consider differences in cognitive tasks and the relevant complexity measures for them. For example, Szymanik (ibid.) suggests that while model-checking complexity is suitable for measuring difficulty of the cognitive task of sentence verification, other notions of complexity could be used for measuring the cognitive difficulty of other processes, such as reasoning.[13]

Since the particular area of cognitive processes this paper focuses on concerns mathematical problem solving, we need to assess whether model-checking complexity is indeed a suitable complexity measure for mathematical cognition. The first potential problem is that mathematics (for the most part) is concerned with *infinite* structures whereas descriptive complexity deals with finite structures, which have a finite universe. This way, descriptive complexity can be treated as a subfield of finite model theory (Szymanik 2016, 103-104), which is hardly representative of mathematics in general. However, we are not here interested in the general question whether a mathematical model $\boldsymbol{A}$ satisfies a sentence ϕ of a logic $\mathscr{L}$. As mentioned in the beginning of this section, we are ultimately interested in *cognitive* complexity,

[13] There are also many other notions of complexity that are potentially relevant to the present topic. Kolmogorov complexity, for example, refers to the length of the shortest computer program that has an informative object, such as a string of symbols, as its output. This notion of complexity seems fitting for mathematical problem solving, but it has turned out that determining the Kolmogorov complexity of even short strings of symbols is a highly difficult task (Soler-Toscano et al. 2014). Other interesting notions of complexity come from machine learning theory, where statistical complexity measures like Gaussian complexity, Rademacher complexity, and Vapnik-Chervonenkis (VC) dimension are used (see Shalev-Shwartz and Ben-David (2014) for more).

i.e., the complexity of a computational model of some cognitive entity. Perhaps it is illuminating to think of such computational models as systems of artificial intelligence (AI). If an AI models a human cognitive process, we can use the algorithmic complexity of the AI to characterise the cognitive complexity of the human process. Since all human cognitive processes are obviously finite, we can limit our considerations to finite computational models. This way, the fact that descriptive complexity is restricted to finite structures is not a problem for the present topic.[14]

However, the above remark of Szymanik suggests another potential problem. Since the field of study in the present context is mathematical problem solving, the complexity measure we use should be suitable for mathematical cognitive processes. However, at first look sentence verification does not appear to be typically the kind of cognitive task mathematicians are involved with. Instead of verifying sentences in models, mathematicians try to construct *proofs*. Would another complexity measure, perhaps one based on satisfiability, be more suitable for the cognitive task of proving theorems?[15] This is a legitimate question to ask and I do not want to claim that descriptive complexity is the most suitable measure generally for mathematical cognitive tasks. Indeed, since I will ultimately argue that results connecting descriptive complexity and computational complexity measures should *not* be used as any kind of general principles when discussing the computational modelling of cognitive capacities, my purpose is not to defend generally the use of descriptive complexity in characterising mathematical (or indeed other) cognitive tasks.[16]

Nevertheless, I can envision how descriptive complexity could be seen to be relevant for the question of complexity of cognitive processes involved in mathematical tasks. Taking a (proposed) computational model M of a mathematical cognitive capacity, we can reasonably ask whether for M some mathematical statement ϕ is part of the output. Furthermore, it is reasonable to ask which logical languages do statements like ϕ in a particular model belong to. Thinking of the computational model again as an artificial intelligence, we can ask whether it models human cognitive capacities with regard to a mathematical task by asking whether it provides as output the same mathematical statements as a human mathematician (given the same input). This way, it is possible to understand the question of computational modelling of human mathematical capacities in the context of model-checking problems, even if we do not believe that human mathematical problems standardly are model-checking.

[14] This is not to say that AI mathematical problem solving should generally aim to model human cognitive processes. Indeed, computer-assisted solutions to mathematical problems tend to take advantage of the brute computational power of computers rather than look for "human-like" proofs. A good example of this is the computer-assisted proof of the four color theorem (Appel and Haken 1976). In modern AI research in mathematics, there are efforts to use statistical "deep learning" to combine with symbolic processes to find new ways of using AI. See, e.g., Lample and Charton (2019). I thank an anonymous reviewer for this last suggestion.

[15] I am grateful to an anonymous reviewer for pointing out this potential difficulty.

[16] This is not to deny that descriptive complexity can be more suitable as a measure for some cognitive tasks and less suitable for others.

I want to emphasise that for various reasons, which will become apparent in the rest of this paper, I do not advocate the above line of reasoning when it comes to computational modelling. Generally speaking, the connections and differences between different complexity measures and different notions of complexity are a much more complicated topic than there is space to describe here, and I do not want to suggest that the simplistic way above of connecting model-checking problems to mathematical cognitive tasks is valid. Nevertheless, I believe that it carries enough force to require a closer analysis of the connection between descriptive complexity and computational complexity, particularly in the context of computational modelling of cognitive capacities involved in mathematical tasks. Furthermore, as we will see, such a closer analysis is fruitful in explicating the weaknesses of applying descriptive complexity measures in the context of mathematical cognition. Let us therefore continue, for now, with the assumption that descriptive complexity measures can be used in characterizing mathematical cognitive tasks.

4 Descriptive Complexity and the Dual Foundations of Mathematics

If we accept the applicability of descriptive complexity in modelling mathematical cognition as described in the previous section, results from the study of descriptive complexity appear to put us in an uncomfortable position. When subscribing to a strong form of tractable cognition restriction, such as the P-cognition thesis, the conclusion seems to be inevitable: at the very least, second-order logic with a least fixed point operator (since it yields the complexity class **EXP**) is intractable. As such, according to the P-cognition thesis, it could not feasibly work as the logic of computational models of human cognitive phenomena. The consensus view implies that the same is true of existential second-order logic, since it is connected with the complexity class **NP**. Only systems of first-order logic yield the complexity class **P** or one of its subclasses and thus can be considered to be computationally tractable. But first-order systems are too weak mathematically to express important properties, such as the least upper bound property for sets of real numbers or, as in our example earlier, the difference between the cardinalities of sets of natural numbers and real numbers.

Therefore we arrive at a fundamental tension. When we consider the connection between mathematics and logic since at least Frege (1884) and Russell (1903), logic has been thought to provide two types of foundations for mathematics. First of these is expressing mathematical concepts in a system of formal logic. The second foundation comes from logic being formalisation of mathematical *thought*. In Frege's (1884) approach, formal logic was meant to provide universally acceptable rules of human thought (free from what he considered to be the arbitrariness of psychologism) that would justify mathematical principles.

However, the complexity considerations above seem to imply that these two foundational roles are in conflict. In order to express many important mathematical concepts, we need to have a system of logic stronger than classical first-order logic.[17] But this connects the required system of logic to a computational complexity class that is considered to be intractable. If we accept the P-cognition thesis, this means that the system of logic used in mathematics can express functions that are too complex to feasibly model human mathematical cognitive capacities. Thus the problem that follows from incorporating the P-cognition thesis, descriptive complexity, and logical foundations of mathematics is as follows: either our logical system is too weak to express important mathematical concepts, or else it is too complex cognitively. This suggests that we need to give up one of the criteria of Frege and Russell. Yet giving up either one is an unappealing prospect. Certainly we do not want to give up important mathematical results because the logical systems used in proving them are considered to be computationally intractable. But from the epistemological point of view, it would seem to be equally problematic to have mathematics built on a system of logic that is considered to be prohibitively complex cognitively.

This prohibitive cognitive complexity can be understood in two different ways. First, we can analyse the strength of a system of logic in terms of its descriptive complexity and connect this to computational modelling of cognitive capacities, as I have been describing above. In this approach, it can be argued that second-order logic cannot be the cognitive basis of mathematics because in it we can express functions that are computationally intractable, which are thought to be unfit to work as computational models. This follows from a direct application of the P-cognition thesis.

It should be noted that the above line of reasoning does not imply that functions expressed in second-order logic are by necessity prohibitively complex. Obviously not everything expressible with second-order logical systems is computationally intractable. For example, an important part of what is expressible in second-order logic is also expressible in first-order logic. But by introducing second-order logic into computational modelling, we are expanding the domain of potential functions that model human cognitive capacities beyond the complexity class **P**. Without further restrictions, this goes against the P-cognition thesis.

The second way to understand the intractability of second-order logic is based on its ability to express and/or decide the truth-value of logical formulas that are too complex for human cognizers to process. This is based on the simple observation

[17] This should not be confused with first-order theories that quantify over second-order objects as providing a basis for mathematics. Most notably this means first-order set theory (ZFC) which has very similar expressive power to second-order logic (Väänänen 2001). It should also be noted that there are first-order systems of logic, such as the independence-friendly logic of Hintikka and Sandu (1989) and the dependence logic of Väänänen (2007), that can express mathematical concepts that are beyond standard first-order logic. These systems don't have contradictory negation so they are not classical. Importantly for the present purposes, both of the above-mentioned systems have equal expressive strength to existential (classical) second-order logic, thus yielding the complexity class **NP**.

that the cognitive process of deciding the truth-value of a logical formula cannot, by definition, be *less* complex than an optimal algorithm for determining the truth-value run by a Turing machine.[18] As explained in Sect. 2, the computational complexity of decision problems, and hence also the corresponding logical formulas, is defined through optimal algorithms for solving them. There are many ways in which the human problem solving algorithms can be computationally *suboptimal* (more on this in the next section), but they cannot outperform optimal algorithms.

For these reasons, it may appear that we cannot escape the problem that the logic we need for expressing familiar mathematical concepts can be, following the P-cognition thesis, prohibitively complex for the modelling of human cognitive capacities. As explained above, this can be understood in two ways, both of which end up with the same problem for the proponents of the P-cognition thesis.[19]

5 Tractable Cognition Thesis Reconsidered

How can we solve the problem presented at the end of the previous section? How can computationally intractable systems of logic work as foundations of mathematics if they are considered to be unfit for modelling human cognitive capacities? In this section I will argue that this is in fact a pseudo-problem that is the result of unwarranted application of results from the study of computational complexity in the domain of computational modelling of cognitive processes. In particular, I question the general use of tractable cognition theses like the P-cognition thesis.

This could seem like a problematic solution. After all, it may appear obvious that *some* form of tractable cognition thesis must be acceptable. Even though the brain is a highly complex organ with a great deal of computational power, it quite clearly has limits. Beyond some limit, computational tasks will be too complex for the brain to carry out. I am not contesting that. What I do want to contest is whether the computational complexity classes, in the present context connected to the descriptive complexity of logical systems, can be used in the general way they are done in the P-cognition thesis. I accept that second-order systems in logic are indeed computationally intractable, but I argue that this notion of computational intractability must be applied in combination with considerations on the kind of algorithms and inputs that are relevant to human cognitive processes.

[18] To be precise, it is the *algorithm* that computes the values of a function modelling the cognitive process that cannot be less complex than an optimal algorithm.

[19] Here I am only considering logic as formalisation of (part of) human *mathematical* cognition, although a more general case could be made that logic should work as the formalisation of universal laws of thought, in the tradition following Boole (1854) and Russell (1903). This would bring in many difficult questions, concerning the prescriptive versus descriptive role of logic, possible cultural differences in logic, etc. Of course such questions are also relevant in the case of mathematical thinking, as well. However, mathematics as a case study is less problematic since there is a wider (although not full) consensus of how mathematical thinking should be formalised with shared logical rules. But this should not be confused with these rules being universal to all humans and not (at least partly) culturally determined (see Pantsar 2014, 2019a for more).

To see why, we need to make a few important clarifications. The first thing to remember is that not all problems in complexity classes like **EXP** (or **NP**) are intractable. Rather, it is the class of **EXP**-*complete* (or **NP**-complete) problems that are considered to be intractable, i.e., those problems that are in **EXP** (or **NP**) but not in **P**. In the case of **NP**, it is of course possible that this class is empty. But even if that were not the case, the question is not about all the problems of **NP**; it is about the so-called ***NP-hard*** problems, referring to those problems that are at least as hard to solve as the hardest problems in **NP**.

However, even with this clarification, there seems to be something weird going on. Recall how **P** and **EXP** provided us with reference points for tractable and intractable algorithms, respectively, the difference being that algorithms in **P** have upper bounds for running time that are polynomial functions of the input while algorithms in **EXP** are lower-bounded by exponential functions. But these may seem like rather useless criteria. We can have, for example, an algorithm that runs for $n^{9999999999999999}$ computational steps, which is sure to take longer than the $2^{n^{1/3}}$ computational steps for prime factoring for any humanly feasible input size n. According to Cobham's thesis, however, the former algorithm is tractable while the latter is intractable. As Aaronson (2012) points out, this characterisation of tractability is accepted in computer science mainly because empirical evidence shows us that it works most of the time. In practice, polynomial time and exponential time have proven be good characterizations for what is considered to be computationally tractable and intractable, respectively.

However, the practice of computer science is potentially very different from the study of tractable *cognition*. Because of this, we must first put tractable cognition theses in their proper place philosophically. There are two ways in which this should be done. The first of these concerns input sizes. We must acknowledge that the complexity classes and the resulting theses of tractability are practical guidelines in theoretical computer science. In particular cases, however, problems in **P** can for inputs up to a certain size be more complex to solve than **EXP**-complete problems. This can continue for a long time as the input sizes grow, but ultimately it will change. It is crucial to remember that the computational complexity measures are asymptotic and measure the complexity of the problem solving task as the input sizes approach infinity. In computer science, the asymptotic complexity measures work in practice, even though problem solving with physical computers is obviously always limited to finite inputs. But the kind of finite inputs computers deal with can be very different from inputs that human problem solvers (without the help of computers) can process. To give just one example, the Goldbach conjecture has been verified for integers up to $4 \cdot 10^{18}$ (Oliveira e Silva et al. 2014).[20] If a printout of an integer has 4,000 digits per page, the largest integer alone would take a thousand *trillion* pages to print. It is obvious that simply reading the integer that is tested would take a prohibitive time for a human problem solver, and thus input sizes of that magnitude

[20] The Goldbach conjecture states every even integer greater than 2 can be expressed as the sum of two primes.

cannot be relevant when we consider the complexity of the cognitive processes involved in the treatment of the Goldbach conjecture.

Based on such differences in relevant input sizes, we must be careful about the conclusions we draw based on the computational complexity measures. Even when limiting the consideration to finite inputs, the fact that the asymptotic measures work as a good practical guideline in computer science does not necessarily imply that they provide similar guidelines for studying the complexity of human cognition. When it comes to human mathematical cognition, the relevant input sizes can be very different from what is feasible for computers. As a consequence of this, for humanly relevant input sizes an **NP**-complete or **EXP**-complete problem can still be solvable. With this sensitivity to bounded inputs, there is nothing to suggest that a function in **NP** or **EXP** could not model the human cognitive process, which goes directly against the P-cognition thesis.

This insight transfers directly to descriptive complexity. It is clear that there are limits to the kind of solutions to mathematical problems that human problem solvers can feasibly provide. If the length of a second-order sentence is beyond the range of what humans can feasibly process, there is no need to consider it relevant for the question what the strength of logical systems can be in order for them to be processable by human cognitive capacities. Based on this reasoning, if the inputs are bounded to humanly relevant sizes, second-order systems could very well be computationally tractable.[21]

Aside from considerations on input sizes, another problematic aspect in applying computational complexity measures to characterise cognitive processes is the fact that the complexity classes are based on *optimal* algorithms for solving problems. As I have argued in (Pantsar 2019b), in mathematics human problem solvers use many heuristic and didactic methods (e.g., diagrams) that involve suboptimal problem solving algorithms. In (Fabry & Pantsar 2019), we argue that mathematical problem solving is a culturally shaped ability that is tightly connected to spatial manipulation of symbols and other ways of engaging with cognitive tools in the problem solving process. Thus the computationally optimal algorithms that are standardly studied in the research of computational complexity can be a bad fit with modelling human problem solving algorithms.[22]

Thus, both in terms of the problem solving algorithms and the input sizes, the standard computational complexity measures fail to be sensitive to important

[21] It has been argued that limiting input sizes like this is not without its problems, since that the total size of input for cognitive capacities can be very large in real life situations, as opposed to the small domains used in studies conducted in lab settings (see, e.g, Van Rooij et al. 2019, 201). While I agree that this is a legitimate concern, I do not see it as an argument against introducing limits to relevant input sizes. Rather, I see it is as a challenge to investigate what the relevant input sizes are for particular cognitive capacities. Obviously there have to exist some limits due to the physiological limitations of our sensory nervous system, and these limits can be enough to make the asymptotic complexity measures misleading or downright irrelevant for that cognitive capacity.

[22] The positive proposal in Pantsar (2019b) and Fabry & Pantsar (2019) is that we should study *humanly optimal* algorithms that are the result of the process of *enculturation*, which refers to the transformative process in which interactions with the surrounding culture determine how cognitive practices are acquired and developed (Menary 2015; Fabry 2020; Pantsar 2019a, 2020).

characteristics of cognitive processes as carried out by human agents. When considering the descriptive complexity of systems of logic, the same conclusions apply. General principles, like the P-cognition thesis, that draw directly from the computational complexity measures are simply too coarse to be applied as reliable theses in cognitive modelling. This does not imply, however, that no form of a tractable cognition principle could be used. I am even ready to agree with Szymanik and Verbrugge (2018) that the P-cognition thesis can provide "a fruitful lens for assessing cognitive tasks". In general, I share their approach that computational complexity measures can provide important tools for the cognitive sciences. Specifically to mathematical problem solving, it is clear that due to physiological limitations there exists a class of problems too complex for human beings to solve as the inputs grow large enough.[23] This class could feasibly intersect in a significant way with the class of **EXP**-complete or **NP**-complete problems. I do not want to claim that the computational complexity measures cannot provide useful information, either generally in computational modelling or specifically in the case of mathematical problem solving. I only argue that they cannot be connected to limits in the modelling of cognitive tasks as directly as is done in the case of P-cognition thesis.

In this way, the problem is not only that the P-cognition thesis appears to be too strict. More generally, limits in computational modelling based directly on computational complexity classes should be treated at best as guidelines. The reason for this is that both in terms of input sizes and the applied algorithms, human problem solving processes can greatly differ from those studied in computer science, for which the complexity classes and the Cobham-Edmonds tractability thesis are meant to apply. In addition to the humanly optimal algorithms as studied in (Pantsar 2019b) and (Fabry & Pantsar 2019), this should prompt us to consider bounded inputs, limited to humanly relevant input sizes.[24]

One such alternative to the P-cognition thesis has been suggested by van Rooij (2008; see Van Rooij et al. 2019 for more details), who argues that by introducing suitable parameters to the computational problem, also super-polynomial time computation can be feasible in modelling cognitive tasks. Van Rooij calls this the "FPT-cognition thesis", for *fixed-parameter tractable*. The introduction of such parameters

[23] With computer assisted problem solving, for example, this class of problems is of course different, which brings us to the important question just what should be included in "human" problem solving. The introduction of cognitive tools, such as pen and paper, abacus, calculators and computers, clearly complicates the matter significantly. Strictly speaking, since computers are human creations, all mathematical problem solving can be seen as human mathematical problem solving. On the other hand, to limit human problem solving to situations where no tools are used (not even pen and paper) would seem needlessly limiting. It is not possible here to go into details, but I believe that there can be a meaningful characterisation of "human" problem solving, based on the idea that a human cognitive process must be essential to reaching the solution. This would not be the case, for example, in simply typing an input to a computer program and reading an output (see Pantsar 2019b for more).

[24] See Buijsman & Pantsar (2020) for a suggestion how to find a more sensitive complexity measure in the case of mental arithmetic, which includes only small input sizes. Instead of being based exclusively on the input size, the complexity measure we suggest is also sensitive to features that have been empirically confirmed to differ in terms of cognitive complexity (based on different reaction times), such as additions involving zero (see, e.g., Brysbaert et al. 1998).

is studied in computational complexity theory (see, e.g., Downey and Fellows 1999) and it can provide more fine-grained complexity considerations to apply to studying the complexity of cognitive tasks. The parameters van Rooij and colleagues discuss are different from limits to input sizes. One example of an NP-hard problem that can be made tractable by fixed parameters is the *minimum vertex cover* problem in graph theory (Van Rooij et al. 2019, 109). A vertex cover of a graph refers to a vertex set that includes (at least) one endpoint of each edge of the graph. If we limit the size of the vertex cover, we get a parametrised version of the minimum vertex cover problem. Many other parameters can also be used, like the maximum degree of a vertex, etc. Note that this is different from imposing limits to input sizes (which in this case would mean a limit to the size of graphs) since the parameter adds another variable, in addition to the input size, into the analysis (Van Rooij et al. 2019, 202). Thus the difference is that parametrisation determines how the complexity of a function is generally characterised (ibid.), whereas limiting the input sizes simply ignores the complexity of a function after some point in the growth of the input sizes. Of course limiting the input size makes intractable functions fixed-parameter tractable (for some value of the parameter), so the two approaches (FPT-cognition thesis and bounded input sizes) are compatible.[25]

Another alternative for the P-cognition thesis is proposed by Szymanik (2016), who argues that the relevant complexity class as a limit for cognitive modelling is, at least in some cases, in fact **NP**.[26] Based on Fagin's theorem and Ristad's (1993) contention that (human) language computations are NP-complete, he formulates the so-called *Ristad's thesis*, according to which everyday language is semantically bounded by properties expressible in existential second-order logic (Mostowski and Szymanik 2012).[27] Szymanik then argues that while such languages contain computationally intractable expressions for human agents, there can be *indirect* mechanisms that make them tractable (in the sense of model checking), meaning that that they can be verified through inferential dependencies with other sentences (Szymanik 2016, 14-16). While Szymanik is concerned with natural languages, Hintikka (1996) has famously argued that his independence friendly (IF) logic can provide a foundation for mathematics. Since IF logic has the same strength as existential second-order logic (Väänänen 2007), combining the views of Szymanik and Hintikka would entail that the foundations of mathematics could be tractable. Hintikka's contention, however, is not generally accepted and it is unclear whether Szymanik's view extends to mathematical inferences. As Szymanik himself notes (2016, 14), there are mathematical expressions like "there exist at most countably many" that are not definable by existential second-order formulas, and would thus require a logical language yielding a computational complexity class greater than **NP**.

[25] See van Rooij et al. (2019, Chapter 5) for details on fixed-parameter algorithms and proving that they are indeed fixed-parameter tractable.

[26] See also Szymanik and Verbrugge (2018), in which they argue for a combination of P-cognition thesis and Fixed-parameter Tractability for NP-complete tasks as a platform for assessing cognitive tasks.

[27] "Everyday language" refers to the *pretheoretic* part of natural language. The reason for this specification is that natural languages (which Ristad writes about) can contain technical expressions that are prohibitively complex (Szymanik 2016, 14).

One of the reasons why Szymanik (2016) eases the relevant complexity class for tractability considerations from **P** to **NP** is that NP-complete problems that are intractable to solve can have solutions whose correctness can be tractably checked. This is important to remember and it is closely connected to the present considerations on what kind of processes mathematical cognition consists of. In Szymanik's (2016, 10) formulation, we can intuitively think of NP-hard problems as ones that do not have "snappy" algorithmic solutions. Problems become prohibitively complex because the only method for solving them is through brute force by going through all possible combinations. But quite clearly this is not how mathematical problems are solved by human agents, aside from particular cases involving small domains. This is consistent with the way I have argued the P-cognition thesis to be misguided: limits on relevant input sizes and considerations on human problem solving algorithms require thinking of mathematical problem solving in a different context, one in which computational complexity classes are not applied directly. As I see it, this is in line with Szymanik's argumentation concerning everyday languages, and there is no reason to believe logical languages connected with complexity classes greater than **P** to be intractable in the relevant human sense. However, while Szymanik argues that the complexity class **NP** provides a relevant limit for language processing in everyday languages, for mathematical languages I do not see any *a priori* reason to use even **NP** as a limit of complexity. It is only after considering the relevant human problem solving algorithms and input sizes (or fixed parameters) that we should consider the computational complexity classes, and the descriptive complexity of logical languages.[28]

Finally, it should be noted that the present approach is very much in line with how computational-level theories are actually used in cognitive science. Generally speaking, theories are not rejected even though they are known to be intractable (usually **NP**-hard, but possibly also more complex). Van Rooij and colleagues (2019, 170) list numerous such theories in the literature, ranging from grammar processing (Ristad 1993; Berwick et al. 1987) to bottom-up visual matching (Tsotsos 1990). In the practice of cognitive science, tractability considerations are often ignored, which goes against principles like the P-cognition thesis and the theoretical discussions on them. Based on the considerations in this paper, however, there is not necessarily anything problematic in that. This does not mean that computational complexity is not relevant in the computational modelling of cognitive capacities. As detailed in Van Rooij et al (2019, Chapter 8), computational intractability can be put into use in many ways in revising computational models of cognitive processes. Along similar lines, instead of dismissing the importance of complexity considerations, I have

[28] For example, as mentioned earlier, second-order logic with a least fixed point operator yields the complexity class **EXP**. Under restrictions for humanly relevant input sizes, however, there is no *a priori* reason to believe that this logic is prohibitively complex. In mathematical practice, second-order logic is freely used and least fixed point operators play a crucial role in recursive definitions. Outside the field of pure mathematics, there are many examples of problems generally considered to be EXP-complete. One of the first of these presented in the literature was finding a perfect strategy in generalised $(n \times n)$ chess (Fraenkel and Lichtenstein 1981).

argued in this paper for carefulness and context-sensitivity in making connections between computational and cognitive complexity.

6 Conclusion

Considering descriptive complexity, the upshot of the considerations in the previous section is clear. If the tractable cognition theses based on computational complexity are translated into the framework of descriptive complexity, it would mean that second-order logic would be (at least partly) unfit as a logical system used in modelling mathematical cognitive capacities, or as a logical system thought to be processable by our cognitive capacities. This would be definitely so if it includes a least fixed point operator (since it yields the complexity class **EXP**), and likely so already in the case of existential second-order logic (which yields the complexity class **NP)**. This is a troubling prospect, since it would imply that we use in mathematics a system of logic that cannot feasibly work as the basis for the cognitive modelling of mathematical thinking.

Fortunately the matter changes fundamentally if we limit our considerations to humanly relevant input sizes and humanly relevant problem solving algorithms. There are many good reasons for doing that, but one very basic reason is already that processing input always takes time and the longer a logical formula, the more time it takes to process it as input. Here I do not want to suggest any particular limits, because with cognitive tools humans can increase the speed of processing input. Perhaps an even more relevant limit is what human beings can process in their working memory, which is another aspect of cognition that can be enhanced with cognitive tools. Nevertheless, there is always some limit after which the input simply takes too much time to process or the memory load becomes too large. In the simplest case of reading a logical formula, for example, it makes no sense to consider cases where it would necessarily take hours, days, or even years to complete the process of feeding in the input. In computer-assisted problem solving the limits are different, but for every process some such limit to the input size can be established.[29]

Based on such considerations, I have argued that any philosophical treatment of tractable cognition needs to include considerations on aspects of human mathematical problem solving that the computational complexity measures are not sensitive to. In doing this, I am not claiming that the P-cognition thesis, or another complexity principle, could not be along the right lines for an important amount of cases. It could indeed be a useful guideline to focus on functions computed by P-hard algorithms as models of human cognitive capacities. But this must not be considered to be a strict criterion.

I have argued that based already on the question of humanly relevant inputs, in addition to the computational complexity of a function, also its behaviour for different input sizes needs to be studied. If the values of a function start to become

[29] This kind of complexity could be better assessed through the notion of *expression complexity* discussed in Sect. 3, which measures complexity with regard to the size of the formula and not the model.

prohibitively complex to compute only for humanly unfeasible input sizes, there is no *prima facie* reason why the function could not work as a model of a human cognitive phenomenon. In moving the focus to descriptive complexity, the same conclusion applies for formulas in supposedly intractable logical systems, such as full second-order logic. Together with the other problems I have discussed in this paper, it becomes clear that applying principles based on descriptive complexity measures in the cognitive sciences is no less problematic than applying computational complexity classes in the manner of the P-cognition thesis.

This is not to say that they cannot be used fruitfully at all. The complexity of different logical systems can be relevant for a wide field of issues both in cognitive science and philosophy. In the foundations of mathematics, the descriptive complexity measures can also be relevant. An argument could be made, for example, that the underlying logic of mathematics should not be any more complex computationally than necessary. As a general guideline, it is feasible that the less complex a logical language is computationally, the less complex it is also cognitively. But already at that point we must be very careful to consider complexity in a way that is relevant to the actual cognitive processes of human agents: the kind of algorithms they use and the types of inputs they process. Only then can we assess accurately the question of cognitive complexity of logical languages. In principle, however, there is no reason to believe that, with appropriate restrictions, second-order logical systems could not provide a feasible basis for the computational modelling of mathematical cognition. Consequently, neither is there any reason to believe that we could not have a common logic for the foundations of mathematics and the modelling of mathematical cognition.

Acknowledgements I would like to thank Fausto Barbero, Regina E. Fabry and Gabriel Sandu for very helpful discussions. I would also like to express my gratitude to the two anonymous reviewers, whose comments have greatly improved this paper.

Funding Open access funding provided by University of Helsinki including Helsinki University Central Hospital. This research has been funded by grants from Alfred Kordelinin säätiö and Suomen kulttuurirahasto.

References

Aaronson, S. (2009). BQP and the Polynomial Hierarchy. *Proc. 42nd Symposium on Theory of Computing (STOC 2009)*, Association for Computing Machinery, 141– 150.

Aaronson, S. (2012). Why philosophers and cognitive scientists should care about computational complexity. In J. Copeland, et al. (Eds.), *Computability: Gödel, Turing, Church, and beyond*. Cambridge: MIT Press.

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

Arora, S., & Barak, B. (2007). *Computational complexity. A modern approach*. Cambridge: Cambridge University Press.

Berwick, R. C., Barton, G. E, Jr., & Ristad, E. S. (1987). *Computational Complexity and Natural Language*. Cambridge, MA: MIT Press.

Blum, M. (1967). A machine-independent theory of the complexity of recursive functions. *Journal of the ACM (JACM)*, *14*(2), 322–336.

Boole, G. (1854). An investigation of the laws of thought: on which are founded the mathematical theories of logic and probabilities. *Dover*,. Publications.

Brysbaert, M., Fias, W., & Noël, M. (1998). The Whorfian hypothesis and numerical cognition: Is "twenty-four" processed in the same way as "four-and-twenty"? *Cognition*, *66*(1), 51–77.

Buijsman, S. & Pantsar M. (2020). Complexity of mental integer addition. *Journal of Numerical Cognition, 6*(1), 148–163.

Church, A. (1936). An unsolvable problem of elementary number theory. *American Journal of Mathematics.*, *58*(58), 345–363.

Cobham, A. (1964). The intrinsic computational difficulty of functions, *Proceedings of the 1964 Congress on Logic, Mathematics and the Methodology of Science,* 24–30.

Cummins, R. (2000). How to solve it. "How does it work?" vs. "What are the laws?" Two conceptions of psychological explanation. In F. Keil & R. Wilson (Eds.), *Explanation and cognition* (pp. 117–145). Cambridge, MA: MIT Press.

Downey, R. G., & Fellows, M. R. (1999). *Parameterized complexity*. New York: Springer-Verlag.

Edmonds, J. (1965). Paths, trees, and flowers. *Canadian Journal of Mathematics*, *17*(3), 449–467.

Fabry, R.E. (2020). The cerebral, extra-cerebral bodily, and socio-cultural dimensions of enculturated arithmetical cognition. *Synthese, 197*, 3685–3720.

Fabry, R.E. & Pantsar, M. (2019). A fresh look at research strategies in computational cognitive science: The case of enculturated mathematical problem solving. *Synthese*. https://doi.org/10.1007/s11229-019-02276-9

Fagin, R. (1974). Generalized First-order Spectra and Polynomial-time Recognizable Sets. In R.Karp (ed.): *Complexity of Computation*, SIAM-AMS Proceedings, Vol. 7, 43-73.

Fraenkel, A. S., & Lichtenstein, D. (1981, July). Computing a perfect strategy for n × n chess requires time exponential in n. In *International Colloquium on Automata, Languages, and Programming*. Springer: Berlin, Heidelberg, 278-293.

Frege, G. (1884). *The Foundations of Arithmetic*. Oxford: Basil Blackwell.

Frixione, M. (2001). Tractable competence. *Minds and Machines*, *11*(3), 379–397.

Garey, M. R., & Johnson, D. S. (1979). *Computers and intractability: A guide to the theory of NP-completeness*. New York: W.H. Freeman.

Gigerenzer, G., Hoffrage, U., & Goldstein, D. G. (2008). Fast and frugal heuristics are plausible models of cognition: Reply to Dougherty, Franco-Watkins, and Thomas (2008). *Psychological Review*, *115*(1), 230–239.

Goldreich, O. (2008). Computational complexity: A conceptual perspective. *ACM Sigact News*, *39*(3), 35–39.

Grohe, M. (1999). Descriptive and parameterized complexity. In J. Flum & M. Rodriguez-Artalejo (Eds.), *Computer Science Logic 1999* (pp. 14–31). Berlin, Heidelberg: Springer.

Hemaspaandra, L. (2018). 17.5 Complexity classes. In K.H. Rosen (ed.): H*andbook of Discrete and Combinatorial Mathematics. Discrete Mathematics and Its Applications* (2nd ed.), CRC Press, pp. 1308– 1314.

Hintikka, J. (1996). *Principles of mathematics revisited*. New York: Cambridge University Press.

Hintikka, J., & Sandu, G. (1989). Informational independence as a semantical phenomenon. *Studies in Logic and the Foundations of Mathematics*, *126*, 571–589.

Hodges, W. (1993). *Model theory*. Cambridge: Cambridge University Press.

Horgan, T., & Tienson, J. (1996). *Connectionism and the philosophy of psychology*. Cambridge, MA: MIT Press.

Immerman, N. (1995). Descriptive complexity: A logician's approach to computation. *Notices of the American Mathematical Society*, *42*(10), 1127–1133.

Immerman, N. (1999). *Descriptive complexity*. New York: Springer Science & Business Media.
Isaac, A. M., Szymanik, J., & Verbrugge, R. (2014). Logic and complexity in cognitive science. In *Johan van Benthem on logic and information dynamics* (pp. 787-824). Springer International Publishing.
Lample, G. & Charton, F. (2019). Deep Learning for Symbolic Mathematics. arXiv:1912.01412
Lucas, J. R. (1961). Minds, machines and Gödel. *Philosophy*, 112–127.
Marr, D. (1982). *Vision: A computational investigation into the human representation and processing of visual information*. San Francisco: W.H. Freeman and Company.
Menary, R. (2015). *Mathematical cognition: A case of enculturation. Open MIND*. Frankfurt a. M.: MIND Group.
Mostowski, M., & Szymanik, J. (2012). Semantic bounds for everyday language. *Semiotica*, *188*(1–4), 363–372.
Newell, A. (1982). The knowledge level. *Artificial intelligence*, *18*(1), 87–127.
Oliveira e Silva, T., Herzog, S., & Pardi, S., (2014). Empirical verification of the even Goldbach conjecture and computation of prime gaps up to $4*10^{18}$. *Mathematics of Computation*, *83*(288), 2033–2060.
Pantsar, M. (2009). *Truth, Proof and Gödelian Arguments: A Defence of Tarskian Truth in Mathematics*. Philosophical Studies from the University of Helsinki, Vol. 23, Helsinki: the University of Helsinki.
Pantsar, M. (2014). An empirically feasible approach to the epistemology of arithmetic. *Synthese, 191*(17), 4201–4229.
Pantsar, M. (2019a). The enculturated move from proto-arithmetic to arithmetic. *Frontiers in Psychology, 10*, 1454.
Pantsar, M. (2019b). Cognitive and computational complexity: Considerations from mathematical problem solving. *Erkenntnis*. https://doi.org/10.1007/s10670-019-00140-3.
Pantsar, M. (2020). Mathematical cognition and enculturation: introduction to the Synthese special issue. *Synthese, 197*, 3647–3655.
Papadimitriou, C. (1994). *Computational complexity*. Boston: Addison-Wesley.
Penrose, R. (1989). *The Emperor's new mind: Concerning computers, minds and the laws of physics*. Oxford: Oxford University Press.
Penrose, R. (1994). *Shadows of the Mind. A Search for the Missing Science of Consciousness*. Oxford: Oxford University Press.
Piccinini, G. (2003). Alan Turing and the mathematical objection. *Minds and Machines*, *13*(1), 23–48.
Pomerance, C. (1996). A tale of two sieves. *Notices of the American Mathematical Society*, *43*(12), 1473–1485.
Pylyshyn, Z. W. (1984). *Computation and cognition: Towards a foundation of cognitive science*. Cambridge, MA: MIT Press.
Ristad, E. S. (1993). *The Language Complexity Game*. Cambridge, MA: MIT Press.
Russell, B. (1903). *Principles of Mathematics*. New York: Norton.
Shalev-Shwartz, Shai, & Ben-David, Shai. (2014). *Understanding Machine Learning - from Theory to Algorithms*. Cambridge: Cambridge University Press.
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), e96223.
Stockmeyer, L. J. (1977). The polynomial-time hierarchy. *Theoretical Computer Science*, *3*, 1–22.
Szymanik, J. (2016). *Quantifiers and Cognition: Logical and Computational Perspectives*, Studies in Linguistics and Philosophy Vol. 96, Springer.
Szymanik, J., & Verbrugge, R. (2018). Tractability and the computational mind. In M. Sprevak & M. Colombo (Eds.), *The Routledge Handbook of the Computational Mind* (pp. 339–354). Routledge.
Tsotsos, J. K. (1990). Analyzing vision at the complexity level. *Behavioral and Brain Sciences*, *13*(3), 423–445.
Turing, A. M. (1936). On computable numbers, with an application to the Entscheidungsproblem. *Proceedings of the London Mathematical Society*, *42*, 230–265.
Turing, A.M. (1948). Intelligent Machinery, reprinted in D.C. Ince (1992) (ed): *Collected Works of A.M. Turing: Mechanical Intelligence*, Amsterdam: North Holland, 87– 106.
van Rooij, I. (2008). The tractable cognition thesis. *Cognitive Science*, *32*, 939–984.
Van Rooij, I., Blokpoel, M., Kwisthout, J., & Wareham, T. (2019). *Cognition and Intractability: A Guide to Classical and Parameterized Complexity Analysis*. Cambridge: Cambridge University Press.
Vardi, M. Y. (1982). The complexity of relational query languages, In *STOC'82: Proceedings of the 14th Annual ACM Symposium on Theory of Computing*. New York: ACM Press, 137– 146.

Vollmer, H. (1999). *Introduction to Circuit Complexity. A uniform approach.* Texts in Theoretical Computer Science. Berlin: Springer-Verlag.
Väänänen, J. (2001). Second-order logic and foundations of mathematics. *Bulletin of Symbolic Logic*, *7*(4), 504–520.
Väänänen, J. (2007). *Dependence Logic: A New Approach to Independence Friendly Logic*. New York: Cambridge University Press.
Väänänen, J. (2019). Second-order and Higher-order Logic, in E. Zalta (ed.): *The Stanford Encyclopedia of Philosophy*. Fall 2020 edition. https://plato.stanford.edu/archives/fall2020/entries/logic-higher-order. Retrieved July 30th 2020.