Cognitive Science 32 (2008) 939–984

ISSN: 0364-0213 print / 1551-6709 online
DOI: 10.1080/03640210801897856

The Tractable Cognition Thesis

Iris van Rooij

Nijmegen Institute for Cognition and Information, Radboud University Nijmegen

Received 7 May 2007; received in revised form 26 November 2007; accepted 12 December 2007

Abstract

The recognition that human minds/brains are finite systems with limited resources for computation has led some researchers to advance the *Tractable Cognition thesis*: Human cognitive capacities are constrained by computational tractability. This thesis, if true, serves cognitive psychology by constraining the space of computational-level theories of cognition. To utilize this constraint, a precise and workable definition of "computational tractability" is needed. Following computer science tradition, many cognitive scientists and psychologists define computational tractability as polynomial-time computability, leading to the *P-Cognition thesis*. This article explains how and why the P-Cognition thesis may be overly restrictive, risking the exclusion of veridical computational-level theories from scientific investigation. An argument is made to replace the P-Cognition thesis by the *FPT-Cognition thesis* as an alternative formalization of the Tractable Cognition thesis (here, FPT stands for fixed-parameter tractable). Possible objections to the Tractable Cognition thesis, and its proposed formalization, are discussed, and existing misconceptions are clarified.

Keywords: Cognitive modeling; Computational-level theory; Philosophy of mind; Philosophy of computation; Complexity theory; Intractability; NP-hard; Constraint satisfaction

One of the primary aims of cognitive psychology is to explain human cognitive capacities (Cummins, 2000). Cognitive capacities are often believed to be the result of the human mind/brain's ability to transform certain input states (e.g., sensations, perceptions, and concepts) into certain output states (e.g., inferences, decisions, plans, and overt responses). Cognitive scientists attempt to model such capacities by constructing precise characterizations of the hypothesized inputs and outputs of cognitive capacities as well as the functional mappings between them. This is what David Marr (1982) called the *computational-level theory* of a cognitive process.

A problem faced by cognitive scientists is that computational-level theories are grossly underconstrained by the available empirical data; not only because any finite number of

Correspondence should be sent to Iris van Rooij, Nijmegen Institute for Cognition and Information, Radboud University Nijmegen, B.02.32, Spinozagebouw, Montessorilaan 3, 6525 HR Nijmegen, The Netherlands. E-mail: i.vanrooij@nici.ru.nl

observed input–output pairs is always consistent with different mappings, but even more so because many of the inputs and outputs of interest to cognitive psychology occur inside the minds of people and are thus not directly observable. For this reason, cognitive scientists could greatly benefit from theoretical constraints on the set of feasible computational-level theories (see also Anderson, 1978, 1990).

This article investigates the proposal, actively advanced by a small group of independent researchers (Cherniak, 1986; Frixione, 2001; Levesque, 1988; Tsotsos, 1990) and tacitly accepted by a larger group (e.g., Anderson, 1990; Martignon & Hoffrage, 2002; Martignon & Schmitt, 1999; Millgram, 2000; Oaksford & Chater, 1993, 1998; Parberry, 1997; Rensink & Provan, 1991; Simon, 1957, 1988, 1990; Thagard, 2000; Thagard & Verbeurgt, 1998), that the mathematical theory of NP-completeness (here NP stands for *N*on-deterministic *P*olynomial-time) can assist cognitive science by delivering such theoretical constraints. I analyze the proposal as consisting of two parts. The first part involves an informal thesis that I call the *Tractable Cognition thesis*: Human cognitive capacities are constrained by the fact that humans are finite systems with limited resources for computation. The second part involves a specific formalization of the Tractable Cognition thesis, what I call the *P-Cognition thesis*: Human cognitive capacities are *polynomial-time computable*.

I agree with above-mentioned researchers that cognitive scientists would do well by adopting the Tractable Cognition thesis in one form or another. In fact, I believe the thesis is widely underappreciated and underutilized by cognitive scientists, and much of this article is devoted to strengthening its position. I do have two points of criticism, but neither is aimed at the Tractable Cognition thesis as such. My first point of critique involves the way in which some researchers have been using the Tractable Cognition thesis in a conceptually problematic fashion. In this article, I try to clarify and explain what I take to be the proper use of the thesis. My second point of critique involves the P-Cognition thesis as a formalization of the Tractable Cognition thesis. I argue that the requirement that computational-level theories describe polynomial-time computable capacities may be overly restrictive, risking the exclusion of veridical computational-level theories from scientific investigation. I propose the *FPT-Cognition thesis* as an alternative formalization (here, FPT stands for fixed-parameter tractable). The FPT-Cognition thesis recognizes that not all super-polynomial time algorithms are created equal (Downey & Fellows, 1999; Flum & Grohe, 2006; Niedermeier, 2006), and considers super-polynomial time computation feasible as long as it is confined to small input parameters.

The structure of this article is as follows: To fix intuitions, I start by formally defining what we mean by a cognitive capacity and its computational-level theory (section 1). I then trace the development from the Church–Turing thesis to a first, informal formulation of the Tractable Cognition thesis (section 2). Section 3 presents the P-Cognition thesis and its relevant formalisms. I identify three different uses of the P-Cognition thesis in present-day cognitive science and argue that only one of them is conceptually coherent in section 4. Subsequently, I present arguments for a refinement of the P-Cognition thesis in the form of the FPT-Cognition thesis. Toward this end, I introduce concepts and methodologies from the field of parameterized complexity (section 5). Finally, I discuss a set of potential objections to the Tractable Cognition thesis and its purported application in cognitive science. For each objection I present a brief reply to either rebut or qualify the objection (section 6).

1. Cognitive capacities as mathematical functions

Cognitive capacities come in many kinds and flavors. Some are high-level and/or knowledge rich (e.g., reasoning, decision-making, categorization); others are low-level and/or knowledge poor (e.g., form perception, motor planning). A cognitive capacity can be a sub-capacity (e.g., letter recognition) or super-capacity (e.g., sentence understanding) of another capacity (e.g., word recognition). Further, a cognitive capacity may be realized by a whole organism (e.g., a human), a part of an organism (e.g., a brain, a neuronal group), a group of organisms (e.g., a social group, a society), an organism-tool complex (e.g., a human–computer combination), or even an altogether artificial device (e.g., a computer).[1]

A common assumption in cognitive psychology—and one that we will adopt in this article—is that a cognitive capacity involves the effective computation of a specific input/output function ψ (Anderson, 1990; Cummins, 2000; Massaro & Cowan, 1993): Given an initial or input state, i, the capacity ψ realizes a final or output state $o = \psi(i)$ as specified by a capacity defining function $\psi : I \rightarrow O$. Capacity defining *functions* are often also referred to as *problems*. Because psychologists are interested in cognitive capacities *qua* generic capacities, an input state is usually seen as a particular input state (e.g. a particular tone, a particular word, a particular pattern of neuronal firing) belonging to a general class of possible input states (e.g., all perceivable tones, all English words, all possible patterns of neuronal firing). Formulating a computational-level theory of capacity ψ consists of hypothesizing a domain of inputs on which the capacity operates, $I_T = \{i_1, i_2, \ldots\}$, a relevant range of outputs, $O_T = \{o_1, o_2, \ldots\}$, and a function, $\psi_T : I_T \rightarrow O_T$, mapping each input $i \in I_T$ to an output $o = \psi_T(i)$ (the subscript T indicates that I_T, O_T and ψ_T are the *theorized* input and output domains and mapping between them). The computational-level theory $\psi_T : I_T \rightarrow O_T$ instantiates a *veridical* description of capacity ψ if and only if it hypothesizes the correct input and output domain and the correct mapping between them (i.e., $I = I_T$ and $\psi_T(i) = \psi(i)$ for all $i \in I$). Assuming that ψ_T is veridical at the computational level, further attempts can be made to understand the effective procedure by which $\psi_T = \psi$ is computed (i.e., what Marr, 1982, called the *algorithmic-level* explanation), as well as to understand how that procedure is physically implemented by neural or other bodily processes (i.e., what Marr called the *implementation-level* explanation).

Cognitive scientists and psychologists often attempt to explain a cognitive capacity (e.g., the ability to understand language) by decomposing it into several sub-capacities (e.g., the capacity to parse sentences, the capacity to recognize letters and words, etc.). The coordinated manifestation of the sub-capacities is then believed to amount to the realization of the analyzed super-capacity (Cummins, 2000). Each hypothesized sub-capacity again calls for its own computational-level explanation. Because there are many different ways in which to characterize a capacity—and even more ways to decompose it—psychologists will often be led to modeling (sub-)capacities that do not actually exist. In such cases the function ψ_T does not have a natural referent and is non-veridical by definition. Because modeling a non-existing capacity is a waste of time, an important part of explaining cognitive capacities is figuring out which sub-capacities are real and which are not. A first step in distinguishing real from apparent cognitive capacities is distinguishing *possible* from *impossible* cognitive capacities. If

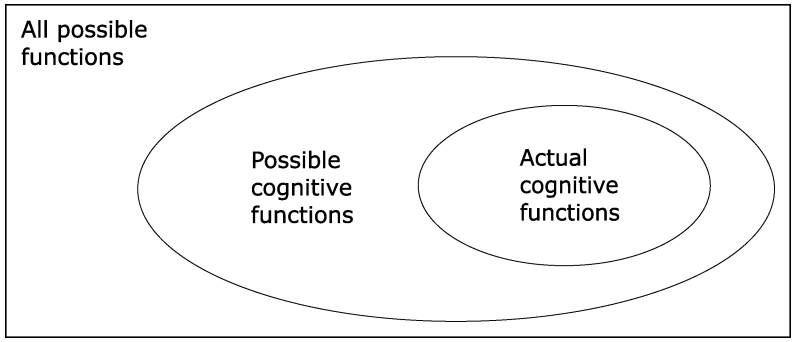

Fig. 1. The set of functions describing *actual* cognitive capacities is a subset of the set of functions describing *possible* cognitive capacities.

one knows a capacity is impossible to realize—for reasons we will pursue in a moment—then one knows it cannot exist as a cognitive capacity (Fig. 1).

2. Conceptual foundations of computability and tractability

2.1. The Church–Turing thesis

The assumption that human cognitive capacities consist in the effective computation of input/output functions naturally imposes a limit on human cognitive capacities. It implies that human cognitive capacities are limited to those input/output functions for which there exist effective procedures. In order to utilize this constraint, we need to make more precise what we mean by an effective procedure. An effective procedure—also called *algorithm*—for a function ψ: $I \rightarrow O$, is a finite step-by-step procedure that when followed mindlessly will transform any input $i \in I$ into the required output $\psi(i)$. The notion of an effective procedure, so defined, is an intuitive notion. Mathematicians and computer scientists have put forth several formalizations (e.g., Church, 1936; Kleene, 1936; Post, 1936), but probably the best-known among cognitive scientists and psychologists is the *Turing machine* formalism (Turing, 1936). In the remainder of this article, we will work with the Turing machine formalism. The reader unfamiliar with this formalism is referred to the Appendix. Other accessible introductions can be found in Frixione (2001), Li and Vitányi (1997), and Wells (1998).

Turing (1936) presented his machine formalization as a way of making the intuitive notions of "computation" and "algorithm" precise. He proposed that every function for which there is an algorithm—which is intuitively computable—is computable by a Turing machine. In other words, functions that are not computable by a Turing machine are not computable in principle by any machine. In support of his thesis, Turing showed that Turing-computability is equivalent to a different formalization independently proposed by Church (1936). The thesis by both Turing and Church that their respective formalizations capture the intuitive notion of algorithm is now known as the Church–Turing thesis. Further, Turing's and Church's formalizations have also been shown equivalent to other accepted formalizations of computation, by which

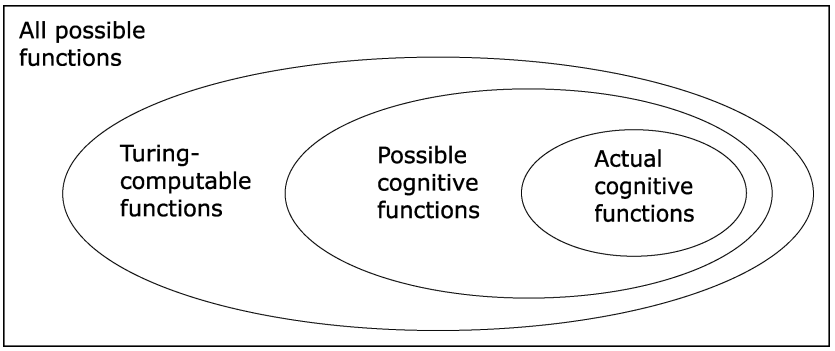

Fig. 2. In the Church–Turing thesis, the set of functions describing *possible* cognitive capacities is a subset of the class of *Turing-computable* functions.

the thesis gained further support (see, e.g., Gandy, 1988; Israel, 2002; Kleene, 1988, for discussions).

Note that the Church–Turing thesis is not a mathematical conjecture that can be proven right or wrong. Instead the Church–Turing thesis is a hypothesis about the state of the world. Although we cannot prove the thesis, it would be in principle possible to falsify it; this would happen, for example, if one day a formalization of computation were developed that (a) is not equivalent to Turing computability; and that, at the same time, (b) would be accepted by (most of) the scientific community. For now, the situation is as follows: Most mathematicians and computer scientists accept the Church–Turing thesis, either as plainly true or as a reasonable working hypothesis. The same seems to be true for many cognitive scientists and psychologists, at least for those who are familiar with the thesis.

The Church–Turing thesis has a direct implication for cognitive psychology: The functions describing possible cognitive capacities are a subset of the class of Turing-computable functions (see Fig. 2). Hence, if the Church–Turing thesis is true, then a computational-level theory that assumes that the cognitive system under study computes an uncomputable function can be rejected on theoretical grounds.

2.2. Criticisms of the Church–Turing thesis

Despite its wide acceptance, the Church–Turing thesis, and its application to human cognition, is not without its critics. The critics can be roughly divided into two camps: those who believe that cognitive systems can do "more" than Turing machines and those who think they can do "less."

Researchers in the first camp pursue arguments for the logical possibility of machines with so-called super-Turing computing powers (e.g., Copeland, 2002; Steinhart, 2002).[2] Much of this work is rather philosophical in nature, and is concerned more with the notion of what is computable in principle by hypothetical machines and less so with the notion of what is computable by real, physical machines (although some may agree to disagree on this point (see, e.g., Cleland, 1993, 1995; but see also Horsten & Roelants, 1995). The interested reader is referred to the relevant literature for more information about the arguments in this camp

(see also Footnote 2 for references). Here, we will be concerned only with the critics in the second camp.

Researchers in the second camp do not doubt the truth of the Church–Turing thesis (i.e., they believe that the situation depicted in Fig. 2 is veridical), but they consider it too liberal to be of practical use for cognitive psychology (e.g., Frixione, 2001; Levesque, 1988; Oaksford & Chater, 1993, 1998; Parberry, 1994; Simon, 1957, 1988, 1990; Thagard, 2000; Thagard & Verbeurgt, 1998). These researchers argue that humans operate under strong physical constraints (e.g., at most 10^{12} neurons in the human brain, a firing rate of at most 10^3 spikes per second), and thus cognitive capacities are limited to those computations that are computable in a realistic amount of time and with the use of a realistic amount of memory. The study of computational resources, and how they are consumed by computation, is called *computational complexity theory*. It is to this theory that we turn now.

2.3. Computational complexity theory

I introduce here the basic concepts and terminology of computational complexity theory. For more details on computational complexity theory, refer to Garey and Johnson (1979), Karp (1972), and Papadimitriou and Steiglitz (1988).

In computational complexity theory, the complexity of a problem (i.e., input/output mapping) is defined in terms of the demands on computational resources as a *function of the size of the input*. The expression of complexity in terms of a function of the input size is useful and natural. After all, it is not the fact *that* demand on computational resources increases with input size (this will be true for practically all problems), but *how* it increases that tells us something about the inherent complexity of a problem. The most common resources studied by complexity theorists are *time* (how long does it take to compute the input/output mapping) and *space* (how much memory does it take to compute the input/output mapping). Here we will be concerned with *time-complexity* only. This focus on time, rather than space, complexity is natural and justified for our purposes—which is to constrain cognitive theories by upper-bounding their computational resource requirements—because accessing memory takes time and thus the amount of space a computation can consume is upperbounded by the amount of time it can consume (Garey & Johnson, 1979).

I first define the *Big-Oh* notation, $O(.)$, which we use to express input size and time complexity. The $O(.)$ notation is used to express an asymptotic upperbound. A function $f(x)$ is $O(g(x))$ if there are constants $c \geq 0$, $x_0 \geq 1$ such that $f(x) \leq cg(x)$, for all $x \geq x_0$. In other words, the $O(.)$ notation serves to ignore constants and lower order polynomials in the description of a function. For this reason, $O(g(x))$ is also called the *order of magnitude* of $f(x)$. For example, $1 + 2 + \ldots + x = x(x + 1)/2$ is $O(x^2)$, and $x^4 + x^3 + x^2 + x + 1$ is $O(x^4)$. The definition of $O(.)$ can be straightforwardly extended to functions with two or more variables. For example, a function $f(x, y)$ is $O(g(x, y))$ if there are constants $c \geq 0$ and $x_0, y_0 \geq 1$ such that $f(x, y) \leq cg(x, y)$, for all $x \geq x_0$ and $y \geq y_0$.

The notions "input size" and "time complexity" are formalized as follows: If M is a Turing machine and i is an input, then the input size, denoted by $|i|$, is the number of symbols on the tape of M used to represent i. Consider, for example, an input i that is a network of nodes and links. We can encode such a network in several different ways. For example, we can encode it

as an adjacency matrix, with each row r (and column c) of the matrix representing a node in the network, and each cell (r, c) in the matrix coded '1' if, and only if there exists a link (r, c) in the network connecting nodes r and c. Alternatively, we can encode the same network as a list of nodes and links, as follows: $a_1, a_2, \ldots, a_n, (a_1, a_2), (a_2, a_3), (a_1, a_3), \ldots, (a_{n-1}, a_n)$. Let n be the number of nodes in the network and m be the number of links. Then, the size of the matrix encoding is $O(n^2)$ because an $n \times n$ matrix has n^2 cells. The size of the list encoding is $n + m$, which is also $O(n^2)$, because $m < n^2$ and $n + n^2$ is $O(n^2)$.

Using the big-Oh notation and the notion of input size defined earlier, we can now express the time-complexity of algorithms and the input/output mappings that they compute. Let M be an algorithm (e.g., a Turing machine). If for any input i, M halts in at most $O(t(|i|))$ steps, then we say M *runs in time* $O(t(|i|))$. Because we require M to halt in time $O(t(|i|))$ for all possible inputs, $O(t(|i|))$ should be interpreted as an asymptotic bound on M's *worst-case* running time. An input/output function $\psi_T : I \rightarrow O$ is said to be computable in time $O(t(|i|))$, if there exists at least one algorithm M that computes $\psi_T(i)$ for all $i \in I$ such that M runs in time $O(t(|i|))$. If the fastest algorithm computing ψ_T runs in time $O(t(|i|))$, then we say that ψ_T has *time-complexity* $O(t(|i|))$. Often we do not know the fastest algorithm for the function ψ_T. In that case, we use the running time of the fastest *known* algorithm as a measure of time-complexity of ψ_T.

The exact running time $t(|i|)$ of an algorithm may depend on the particular encoding used to represent the input i, and the particular machine model used (Turing machine or otherwise). In what sense, then, is it meaningful to talk about *the* time-complexity of an input/output function? The Invariance thesis provides a possible answer.[3]

The Invariance thesis states that, given a "reasonable encoding" of the input and two "reasonable machines" M_1 and M_2, the complexity of a given computation ψ_T for M_1 and M_2 will differ by at most a polynomial amount (see, e.g., Frixione, 2001; Garey & Johnson, 1979)—that is, if M_1 is computable in time $O(t(|i|))$, then M_2 is computable in time $O(t(|i|)^{\alpha})$ where α is a constant. Here, with "reasonable encoding" is meant an encoding that does not contain irrelevant information and in which numbers are represented in b-ary with $b > 1$ (e.g., binary, or decimal, or any fixed base other than 1). Further, with "reasonable machine" is meant any type of Turing machine (with possible extensions as described in the Appendix section, A.3) or any other realistic computing machine under a different formalization (including, e.g., neural networks, cellular automata). Note, however, that a machine capable of performing arbitrarily many computations in parallel (cf. non-deterministic Turing machine or quantum computer) is not considered a "reasonable" machine (Garey & Johnson, 1979).

Like the Church–Turing thesis, the Invariance thesis is widely accepted among computer scientists and cognitive scientists. The Invariance thesis, if true, implies that we can analyze the inherent worst-case complexity of computations independent of the machine model up to a polynomial amount of precision. In other words, if a computation is of polynomial-time complexity under one model it will be of polynomial-time complexity under any other reasonable model. Similarly, if a problem is of super-polynomial time complexity under one model it will be of super-polynomial time complexity under any other reasonable model.

Although some computer scientists believe that one day it will be possible to build quantum computers (allowing for arbitrarily many parallel computations), this still remains to be seen. Furthermore, even if a quantum computer—or some other computational device of comparable

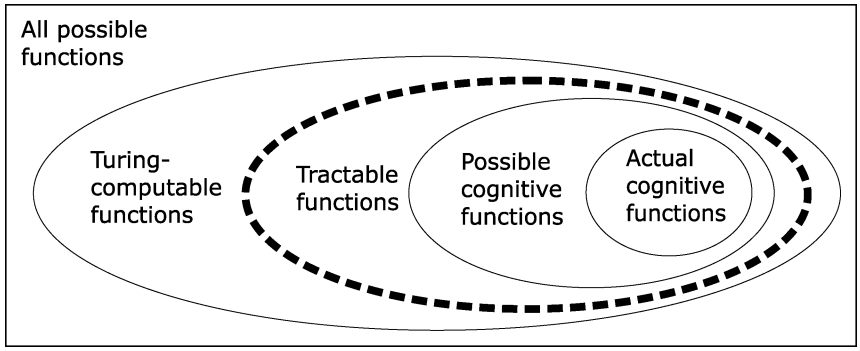

Fig. 3. According to the Tractable Cognition thesis, the set of functions describing *possible* cognitive capacities is a subset of the set of *tractable* functions. The dotted line indicates that the set of tractable functions remains to be formalized.

power—were physically realizable, it still need not be the right model of human computation (see, e.g., Litt, Eliasmith, Kroon, Weinstein, & Thagard, 2006). Whichever will turn out to be the case, for purposes of this article we will adopt the Invariance thesis (see also section 6.8 for a qualification of the computational powers of quantum computers).

2.4. The Tractable Cognition thesis

So far we have talked about "reasonable encoding" and "reasonable machines," but what about "reasonable time?" As discussed in section 2.2, critics of the Church–Turing thesis argued that computability does not pose strict enough constraints on human computation because human cognitive capacities are limited to those that can be computed in a "reasonable time"—called *tractable* computations. This notion of tractability is an intuitive notion, like effectiveness was an intuitive notion before its formalization as Turing-computability. If we could somehow formally define the class of functions that are tractably computable, and the class of functions that are not, then we could have a Tractable Cognition thesis, which states that cognitive functions are among the tractably computable functions. Such a thesis could then serve to provide further theoretical constraints on the set of functions describing human cognitive capacities, as illustrated in Fig. 3.

We do not know exactly what type of machine the human brain, or whatever other physical substrate underlying human cognition, is. Therefore, it would be particularly useful if we could formalize the Tractable Cognition thesis in a way that abstracts away from machine details—that is, like the Church–Turing thesis provides a definition of computability independent of the Turing machine formalization, we would like to have a Tractable Cognition thesis that provides a definition of tractability independent of the Turing machine formalization. From the Invariance thesis (section 2.3) we know that if we could define the set of tractable and intractable problems such that the classification is insensitive to a polynomial amount of complexity in computation of the problem, we would have a machine independent formalization of the notion of (in)tractability.

For now, I have formulated the Tractable Cognition thesis as a "mold" for a formal version of the thesis (indicated by the dotted line in Fig. 3). The remainder of this article investigates two possible formalizations of the Tractable Cognition thesis. The first is called the P-Cognition thesis, and the second is called the FPT-Cognition thesis. We will see that both the P-Cognition thesis and the FPT-Cognition thesis adopt the Invariance thesis to achieve the desired generality and machine independence.

3. The P-Cognition thesis

Because of the Invariance thesis, we can distinguish between functions that can be computed in polynomial time (i.e., time $O(|i|^{\alpha})$) where α is a constant), and those that cannot (e.g., functions that require at best exponential time: time $O(t(\alpha^{|i|}))$ with constant α). The classical[4] theory of computational complexity proposes that the class of functions that are computable in polynomial time provides a useful (approximate) formalization of the set of functions that are intuitively tractable (e.g., Garey & Johnson, 1979; Papadimitriou & Steiglitz; 1988). Garey and Johnson express the intuition as follows:

> Most exponential time algorithms are merely variations on exhaustive search, whereas polynomial time algorithms generally are made possible only through the gain of some deeper insight into the nature of the problem. There is wide agreement that a problem has not been "well-solved" until a polynomial time algorithm is known for it. Hence, we shall refer to a problem as intractable, if it is so hard that no polynomial time algorithm can possibly solve it. (p. 8)

The classical definition of intractability is widely adopted in computer science (see, e.g., Garey & Johnson, 1979; Papadimitriou & Steiglitz, 1988, or any other introductory text). To see that the definition has merit, consider Table 1. Table 1 shows that an exponential function, like $O(2^{|i|})$, grows much faster than a polynomial function, like $O(|i|^2)$. If we assume that a computing machine can compute, say, 100 basic computations per second then, as $|i|$ grows, the exponential running time gets unrealistic very fast for any reasonable computing machine (compare columns 2 and 3 of Table 1).

Here the assumption of 100 basic operations per second is for illustrative purposes only. In the context of cognitive psychology, what constitutes a reasonable assumption depends on the speed of the "hardware" assumed to underly the cognitive capacity under study (e.g., one may assume a generous upperbound for whole brain computational power of about 10^{12} (neurons) $\times$ 10^3 (connections per neuron) $\times$ 10^3 (firing rate) $= 10^{18}$ basic computation steps per second, but presumably not every cognitive capacity can claim all of the processing power of the entire brain). More important, however, increasing the speed with which a basic computational operation can be performed has very limited impact if the running time is of exponential-time complexity. For example, if we could realize a 100-fold speed-up of our computing machine (e.g., by improving its hardware so that it can perform each sequential operation 100 times faster; or by having it run 100 computational channels in parallel), then an exponential-time algorithm running in time $O(2^{|i|})$ would still be impractical for all but relatively small input sizes (compare columns 3 and 5 of Table 1).

Table 1
Illustration of how a polynomial running time, $O(|i|^2)$, and an exponential running time, $O(2^{|i|})$, compare for different $|i|$, assuming either 100 or 10,000 computational steps per second

	Assume 100 steps per second		Assume 10,000 steps per second	
$\|i\|$	$O(\|i\|^2)$	$O(2^{\|i\|})$	$O(\|i\|^2)$	$O(2^{\|i\|})$
2	0.04 sec	0.04 sec	0.02 msec	0.02 msec
5	0.25 sec	0.32 sec	0.15 msec	0.19 msec
10	1.00 sec	10.2 sec	0.01 sec	0.10 sec
15	2.25 sec	5.46 min	0.02 sec	3.28 sec
20	4.00 sec	2.91 hrs	0.04 sec	1.75 min
30	9.00 sec	4.1 months	0.09 sec	1.2 days
50	25.0 sec	8.4×10^4 years	0.25 sec	8.4 centuries
100	1.67 min	9.4×10^{19} years	1.00 sec	9.4×10^{17} years
1,000	2.78 hr	7.9×10^{290} years	1.67 min	7.9×10^{288} years

The classical definition of intractability has found its way into cognitive psychology, leading to the P-Cognition thesis: cognitive capacities are limited to those functions that can be computed in polynomial time (Fig. 4). The P-Cognition thesis is (tacitely) being used by many researchers in cognitive psychology (e.g., Anderson, 1990; Cherniak, 1986; Levesque, 1988; Martignon & Hoffrage, 2002; Martignon & Schmitt, 1999; Millgram, 2000; Oaksford & Chater, 1993, 1998; Parberry, 1997; Simon, 1988, 1990; Thagard, 2000; Thagard & Verbeurgt, 1998; Tsotsos, 1990, 2001), as well as in artificial intelligence (see, e.g., Cooper, 1990; Nebel, 1996); but Frixione (2001) seemed to be the first to explicitly call for its general use in cognitive psychology as a way of constraining the space of possible computational-level theories.

Does the P-Cognition thesis really help as Frixione (2001) suggested? If theories that cognitive psychologists naturally propose are trivially tractable, then the P-Cognition thesis would not help us much. As it turns out, however, many present-day theories are far from (trivially) tractable. Table 2 presents a sample of input/output functions aimed at modeling (hypothesized) cognitive capacities. The first thing to notice is that all these functions have

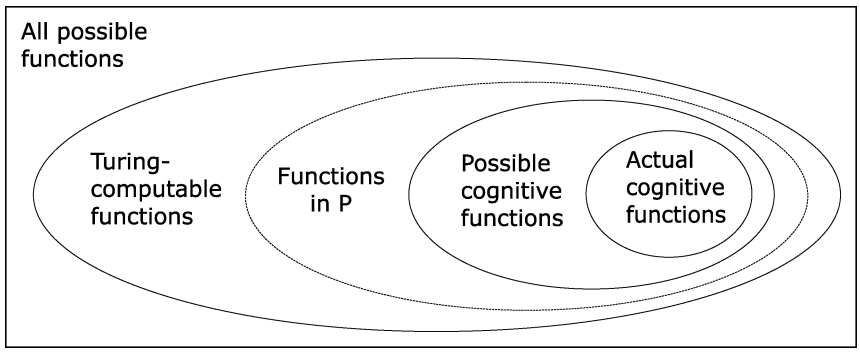

Fig. 4. According to the P-Cognition thesis, the set of functions describing *possible* cognitive capacities is a subset of the set of *polynomial-time computable* functions, P.

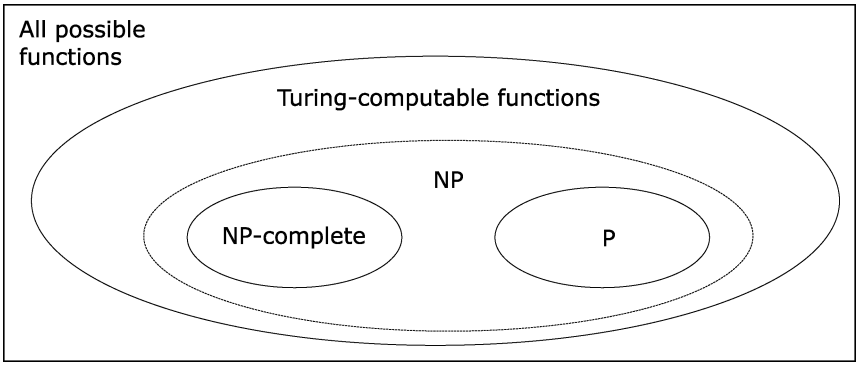

Fig. 5. The view of NP on the assumption that P $\neq$ NP: NP-complete problems are not in P if, and only if, P, $\neq$ NP.

search spaces that are potentially of super-polynomial size. Take, for example, what is often considered the bedrock of cognition: *categorization* (Harnad, 1987). A widely held view is that object categories are formed so as to maximize within-category similarity and between-category dissimilarity (Pothos & Chater, 2001, 2002; Rosch, 1973; Rosch & Mervis, 1975). This idea can be formalized as follows:

> Categorization (ψ_{Cat})
> *Input:* A set of objects, $P = \{p_1, p_2, \ldots, p_n\}$, and a similarity value, $s(p, q)$, and a dissimilarity value, $d(p, q)$, for each $(p, q) \in P \times P$.
> *Output:* A partition of P into k subsets $A_1, A_2, \ldots, A_k$ such that the sum of within-category similarities and between-category dissimilarities is maximized (i.e., the partition maximizes the following sum: $\sum_i \sum_{p,q \in A_i} s(p, q) + \sum_{i,j,i \neq j} \sum_{p \in A_i, q \in A_j} d(p, q)$).

The combinatorial explosion in the number of possible candidate partitions to consider in ψ_{Cat} excludes the possibility of an exhaustive search (even if a human would encounter at most 100 objects in his or her lifetime, there would already be as many as 10^{115} possible partitions to choose from), and surely no psychologist would propose that this is how the brain computes ψ_{Cat}. Nonetheless, proposing ψ_{Cat} as a descriptive model of human categorization raises the question of how the brain would be able to compute it otherwise. To address this question, we need to have a technique that allows us to test whether or not ψ_{Cat} can be computed by a non-exhaustive, polynomial-time algorithm. I will now briefly discuss how the mathematical theory of NP-completeness provides us with such a technique.

The theory of NP-completeness defines two classes of functions (see Fig. 5):[5] Functions that can be computed by a deterministic Turing machine in polynomial time, P, and functions that can be computed by a nondeterministic Turing machine in polynomial time, NP (see the Appendix, section A.3 for an explanation of the difference between the two types of Turing machines). Because every deterministic Turing machine is also a nondeterministic Turing machine, we know that all functions that belong to the class P also belong to the class NP (in other words, P $\subseteq$ NP). It is widely believed that there exist functions in the class NP that do not belong to the class P, and thus P $\subset$ NP. This conjecture is motivated, among other things, by the

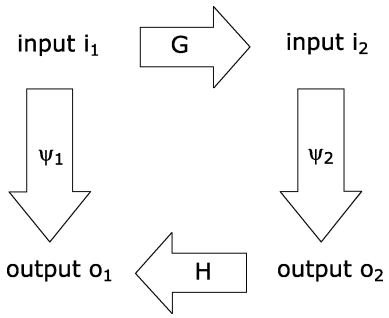

Fig. 6. Illustration of the concept of polynomial-time reduction. The functions G and H together constitute a reduction from ψ_1 to ψ_2 if, and only if, $\psi_1(i_1) = H(\psi_2(G(i_1)))$. In that case, the function $\psi_1(i_1)$ can be computed by first computing $i_2 = G(i_1)$, then computing $o_2 = \psi_2(i_2)$, and finally computing $H(o_2) = \psi_1(i_1)$. The reduction is a polynomial-time reduction if and only if G and H are polynomial-time computable. In that case, ψ_1 is polynomial-time computable if ψ_2 is polynomial-time computable.

existence of so-called NP-hard problems. These problems derive their name from the fact that they *are at least as hard* to compute as *any* problem in the class NP (NP-hard problems that are *in* the class NP are also known as NP-*complete*). To be precise, if any one NP-hard problem were to be computable in polynomial time, then all problems in NP would be computable in polynomial time, which would mean that—contrary to common mathematical intuition—the class P is exactly equal to the class NP. Presently hundreds or thousands of functions are known to be NP-hard (see, e.g., Garey & Johnson, 1979, or available online compendia); but despite great efforts from many computer scientists, nobody to date has succeeded in finding a polynomial-time algorithm that computes an NP-hard function (hence, the belief that P $\neq$ NP). The finding that a function ψ_T is NP-hard is, therefore, seen as strong evidence that ψ_T is *not* polynomial-time computable. In the remainder of this article, we will work under the assumption that P $\neq$ NP.

One can prove a function to be NP-hard using the technique of polynomial-time reduction (see Fig. 6 for an illustration): For two functions, $\psi_1 : I_1 \rightarrow O_1$ and $\psi_2 : I_2 \rightarrow O_2$, we say that ψ_1 *reduces to* ψ_2 if there exist algorithms G and H such that G transforms any input $i_1 \in I_1$ into an input $G(i_1) \in I_2$ and H transforms any output $\psi_2(G(i_1))$ into output $H(\psi_2(G(i_1))) = \psi_1(i_1)$. We furthermore say the reduction is a *polynomial-time* reduction if both G and H run in polynomial time. Now, a function ψ_T is NP-*hard* if every function in NP can be polynomial-time reduced to ψ_T. This technique of polynomial-time reduction is very useful because once a function ψ_1 is known to be NP-hard we can prove another function ψ_2 to be NP-hard by polynomial-time reducing ψ_1 to ψ_2. In 1971, Stephen Cook proved the first function to be NP-hard, called *Satisfiability*, and since then many functions have been shown to be NP-hard by (direct or indirect) reduction from Satisfiability.

We are now in a position to answer our earlier question: Can ψ_{Cat} be computed in a way that avoids an exhaustive search of a super-polynomial (part of the) search space? It turns out that the answer is "no" because the function ψ_{Cat} is NP-hard (e.g., the known NP-hard function Clustering polynomial-time reduces to ψ_{Cat}; Garey & Johnson, 1979). Consequently, *all* algorithms computing ψ_{Cat} will be of super-polynomial time complexity. Notably, the same is

true for many formalizations of the computational-level theories listed in Table 2. It seems then that none of the computational-level theories (trivially) satisfies the tractability requirement posed by the P-Cognition thesis.

4. A critique of the P-Cognition thesis in practice

Being confronted with an NP-hardness result for one's computational-level theory, what is a cognitive psychologist to do? The opinions seem to vary. In the literature at least three different types of responses can be distinguished. Briefly, they are as follows:

1. *Framework rejection response:* Some researchers consider the finding that a computational-level theory ψ_T is NP-hard reason to reject the theoretical framework underlying ψ_T altogether. For example, Oaksford and Chater (1993, 1998) argued that logicists' approaches to modeling common sense reasoning are untenable because checking whether a set of beliefs is logically consistent (i.e., Satisfiability) is NP-hard (cf. Martignon & Schmitt, 1999; Millgram, 2000).
2. *Theory revision response:* Other researchers view the theory of NP-completeness as a tool for refining (not all-round rejecting) computational-level theories such that they satisfy tractability constraints. For example, Levesque (1988), like Oaksford and Chater (1993, 1998), recognized the inherent exponential-time complexity of general logic problems, but unlike Oaksford and Chater, he concluded that we need to adjust logic, not abandon it, in order to obtain psychologically realistic models of human reasoning. Similarly, upon finding that visual search, in its general (bottom-up) form, is NP-complete, Tsotsos (1990, 2001) did not abandon his model of vision, but instead adjusted it by assuming that top-down information helps constrain the visual search space.
3. *Devising heuristics response:* Last, there are researchers who, upon finding that a cognitive function ψ_T is NP-hard, do not reject ψ_T as computational-level theory, nor adjust it to accommodate tractability constraints. Instead they assume that, at the algorithmic level, the function ψ_T is being computed by heuristics or approximation algorithms. This approach is taken, for example, by Thagard and Verbeurgt (1998) in the domain of coherence reasoning (Thagard, 2000; but see also Chater & Oaksford, 2000; Martignon & Schmitt, 1999). It seems that these researchers recognize the constraint that tractability places upon algorithmic-level theories, but consider the constraint irrelevant at the computational level.

Of these reactions, the Type 2 reaction (i.e., *Theory revision*) is the most sensible one. I will explain below why this is so.

An intractability result for ψ_T indicates that it is not (plausibly) possible that the mind/brain computes exactly ψ_T, but it may still compute something quite similar to ψ_T. If a theory ψ_T is found to be NP-hard, a natural next step is to define a new function ψ'_T, such that ψ'_T still captures all that ψ_T was intended to capture, and assess whether or not ψ'_T is tractably computable. If ψ_T was on the right track to begin with, this iterative process will bring us closer and closer to a veridical model of ψ. It is only when very many possible candidate theories for ψ are considered, and none can be shown to be tractably computable, that we

are justified in doubting that our theoretical framework is entirely on the wrong track, and possibly the hypothesized capacity ψ does not exist at all (see section 1).

There are many different ways in which one can attempt to bring a computational-level theory ψ_T within the range of computational tractability, but one obvious approach would be to try and restrict ψ_T to some (psychologically relevant) proper subset $I' \subseteq I$ of inputs, and test whether the resulting function $\psi'_T : I' \rightarrow O$ is computable with a polynomial amount of resources. To illustrate, let us again consider the earlier example ψ_{Cat} as a computational-level model of unsupervised categorization. The input to ψ_{Cat} was defined as follows:

> I_{Cat} : A set of objects, P, and a similarity value, $s(p, q)$, and a dissimilarity value, $d(p, q)$, for each pair of objects $(p, q) \in P \times P$.

Note how $s(p, q)$ and $d(p, q)$ are completely independent measures in I_{Cat}. It may very well be, however, that humans mentally represent similarity and dissimilarity such that the dissimilarity $d(p, q)$ is simply the inverse of the similarity $s(p, q)$. In that case, the input domain of ψ_{Cat} could be naturally restricted as follows:

> I'_{Cat} : A set of objects, P, and a similarity value, $s(p, q)$, and a dissimilarity value, $d(p, q) = -s(p, q)$, for each pair of objects $(p, q) \in P \times P$.

Now we may ask if possibly the restriction $\psi'_{Cat} : I' \rightarrow O$ satisfies the tractability constraint as specified by the P-Cognition thesis.

Importantly, the example that I present is for illustrative purposes only. The restriction of ψ_{Cat} in the form of ψ'_{Cat} may of course fail as an attempt to save the categorization model. This would happen, for example, if the proposed restriction proves indefensible on empirical grounds (see, e.g., Tversky, 1977), or if ψ'_{Cat} is found to be equally intractable as ψ_{Cat}. My point, however, is merely this: The fact that ψ_{Cat} is NP-hard does *not* imply that ψ'_{Cat} is NP-hard. Therefore, the finding that ψ_{Cat} is NP-hard is not sufficient to reject the informal and general hypothesis that categories are formed so as to maximize within-category similarity and between-category similarity, although it is an incentive to reformulate the (formal) computational model ψ_{Cat} in a way that accommodates the requirement of computational tractability.

In general, forcing one's computational model to meet tractability constraints—either by restricting the set of inputs (as illustrated above) or by otherwise adjusting model details (more on this below and in section 5)—ensures that computational-level theories are formulated in a way that leaves open possibilities for algorithmic and implementation level explanation. In fact, the failure to do so can be seen as precluding any understanding of how the capacity is effectively and physically realized given finite resources. This approach conforms to the views of Frixione (2001), Levesque (1988), and Tsotsos (1990), among others, and represents, in my opinion, the proper use of the Tractable Cognition thesis in cognitive psychology.

In the following, I comment on the other two reactions and explain why I think neither is conceptually unproblematic. In particular, I will argue that *Framework rejection* is unjustified because intractability does not propagate to all models in a given generalized class and the *Devising heuristics* response is explanatorily incoherent in the sense that it introduces a logical inconsistency between computational-level explanation, on the one hand, and algorithmic-level explanation, on the other.

The problematic nature of a *Framework rejection* response should be evident from the discussion so far. No single intractability result for ψ_T can overthrow the entire framework in which ψ_T was formulated. As explained above, the finding that a function $\psi_T : I \rightarrow O$ is NP-hard does not imply that a restricted function $\psi_T : I' \rightarrow O$, with $I' \subset I$ is NP-hard (although the reverse is true). This difference between "a general problem being NP-hard" and "a problem being generally NP-hard" seems to be often confused in the cognitive science literature. Consider, for example, the following statement by Thagard (2000):

> Coherence problems are inherently computational intractable, in the sense that . . . there are no efficient (polynomial-time) procedures for solving them [assuming P $\neq$ NP]. (p. 15)

Here, Thagard (2000) based himself on the NP-hardness proof by Verbeurgt (see the Appendix in Thagard & Verbeurgt, 1998) for the general class of functions subsumed by the function in row 3 of Table 2, denoted by ψ_{Coh}.[6] Note how Thagard's (2000) synopsis (misleadingly) suggests that coherence problems are of super-polynomial time complexity *across-the-board*. Clearly, the finding that ψ_{Coh} is NP-hard does not warrant this conclusion. Several of the coherence models that Thagard (2000) discussed in his book are in fact restricted versions or variants of ψ_{Coh}, neither of which automatically inherits the NP-hardness of ψ_{Coh} (see van Rooij, 2003, for a detailed treatment). In a similar vein, Oaksford and Chater (1998) wrote: "Consistency checking constitutes a general class of problems in complexity theory called satisfiability problems" (p. 76) and "consistency checking, like all satisfiability problems, is NP-complete" (p. 77). Here, Oaksford and Chater (1998) based themselves on Cook's (1971) finding that Satisfiability is NP-complete. Their conclusion, however, is as unwarranted as the one by Thagard (2000) quoted above. Surely there exist versions of Satisfiability that are in P (e.g., 2-Satisfiability; Garey & Johnson, 1979). It seems as if these researchers believe that an intractability result for a given model also "infects" all versions and variants of that model. Oaksford and Chater (1998) furthermore took this as a reason to abandon a modeling framework altogether (Thagard, 2000, did not, as his reaction was of the type *Devising heuristics*). Besides being unwarranted, the reaction of Oaksford and Chater (1998) is also counterproductive. Any sufficiently rich framework for modeling cognition will almost certainly give rise to some (possibly many) models that are NP-hard. This is also illustrated by the fact that Bayesianism—the framework that Oaksford and Chater (1998) revert to after rejecting logicism—is similarly plagued by NP-hardness results (Cooper, 1990; Roth, 1996).

In commenting on the *Devising heuristics* response, I distinguish between inexact algorithms with provable approximation guarantees and inexact algorithms without such provable performance. Following computer science convention, I refer to the former as approximation algorithms proper and the latter as heuristics. If we were to accept mere heuristics as algorithmic level descriptions, then there would be no principled relation between descriptions at the algorithmic and descriptions at the computational level. It seems that this would render the two levels of theorizing at best disconnected and at worst inconsistent. To illustrate, let us again consider Thagard's coherence model, ψ_{Coh}. As noted, ψ_{Coh} is NP-hard and Thagard knew this. Instead of reformulating (e.g., restricting) his model, Thagard seemed to take the result as a green card for formulating heuristics "computing" ψ_{Coh} as algorithmic-level descriptions.[7] His favorite heuristic is embodied by a neural network type processing

Table 2
A sample of computational-level theories whose combinatorial search spaces are potentially super-polynomial in $|i|$

Cognitive Domain	Computational-Level Theory (ψ_T)	References
Categorization	*Input:* A set of objects, P, and a (dis)similarity value for each $(p, q) \in P \times P$. *Output:* A partition of P into categories such that within-category similarity and between-category dissimilarity is maximum.	Pothos and Chater (2001, 2002); Rosch (1973); Rosch and Mervis (1975)
Similarity	*Input:* Two objects x and y and a set of transformation rules T. *Output:* The length of the shortest sequence of transformation rules from T that, when applied to x, yields y.	Chater and Vitányi (2003a, 2003b); Hahn, Chater, and Richardson, (2003)
Coherence	*Input:* A set of propositions, P, positive and negative constraints, $C \subseteq P \times P$. *Output:* A truth assignment $T(P)$ satisfying a maximum number of constraints.	Millgram (2000); Thagard (2000); Thagard and Verbeurgt (1998); van Rooij (2003)
Gestalt perception	*Input:* A string s and a decoding function $f : C \rightarrow S$ mapping codes to strings. *Output:* A code $c \in C$ such that $f(c) = s$ and the length of c is minimum.	van der Helm (2004); van der Helm and Leeuwenberg (1986, 1996)
Visual search	*Input:* A target T, a visual display D, and two numbers x and y. *Output:* A subset $S \subseteq D$ such that the number of (mis)matching elements in S and T is at least x (at most y).	Kube (1991); Tsotsos (1990); van Rooij (2003).
Defeasible reasoning	*Input:* A knowledge base K and a set of default rules R. *Output:* All propositions $p_1, p_2, \ldots, p_k$ derivable from K using R, such that $p_1, p_2, \ldots, p_k$ and K are consistent.	Oaksford and Chater (1993, 1998); Reiter (1980).
Bayesian inference	*Input:* A knowledge base K and a set of competing hypotheses H. *Output:* A hypothesis $h \in H$ that maximizes the conditional probability $P(h\|K)$.	Chater, Tenenbaum, and Yuille (2006); Cooper (1990); Roth (1996)
Decision making	*Input:* A set of choice alternatives P and a value function $u : S \rightarrow N$ (where $S \subseteq P$ and N is a set of numbers). *Output:* A subset $S \subseteq P$ such that $u(S)$ is maximum.	Fishburn and LaValle (1993, 1996); van Rooij, Stege, and Kadlec (2005)
Language processing	*Input:* Surface form s, lexicon D, lexical-surface form relation mechanism M. *Output:* Set of lexical forms U generated by D from which M can create s.	Barton, Berwick, and Ristad (1987); Ristad (1990, 1993); Wareham (1996, 1999, 2001)
Planning	*Input:* An initial state s, a goal state g, and a collection of operators O. *Output:* A sequence of operators that when applied to s produces g.	Bylander (1994); Joseph and Plantinga (1985); Newell and Simon (1988a, 1988b)

(Continued on next page)

Table 2
A sample of computational-level theories whose combinatorial search spaces are potentially super-polynomial in $|i|$ (*Continued*)

Cognitive Domain	Computational-Level Theory (ψ_T)	References
Network harmony	*Input:* A harmonic (e.g., Hopfield) neural network. *Output:* An activation pattern that maximizes harmony.	Jagota (1997); Rumelhart et al. (1986); Smolensky and Legendre (2006)
Network learning	*Input:* A neural network N and function f. *Output:* A weight assignment to the connections in N such that N computes f.	Judd (1990); Parberry (1994, 1997)

Note: For brevity, function descriptions have been simplified. See references for more details.

mechanism (see Thagard, 2000; Thagard & Verbeurgt, 1998). Let us call it N_{Coh}. Now there are two possibilities: Either (a) N_{Coh} transforms any input $i \in I_{Coh}$ into $N_{Coh}(i) = \psi_{Coh}(i)$, or (b) there exist some $i \in I_{Coh}$ such that $N_{Coh}(i) \neq \psi_{Coh}(i)$. If case (a) holds, then N_{Coh} is an exact algorithm for ψ_{Coh}, and hence it is not a heuristic. But then, N_{Coh} would be as unrealistic at the algorithmic level as ψ_{Coh} is at the computational level because knowing ψ_{Coh} is NP-hard, the algorithm N_{Coh} would have to run in super-polynomial time. If case (b) holds, on the other hand, then N_{Coh} is a heuristic, and it is possible that N_{Coh} is tractable even if ψ_{Coh} is not. But then, N_{Coh} does not explain how ψ_{Coh} is effectively computed for all $i \in I_{Coh}$ where $N_{Coh}(i) \neq \psi_{Coh}(i)$. This also means that for all these i, $N_{Coh}(i)$ and $\psi_{Coh}(i)$ are competing (and *not* complementary) accounts about what it means for cognizers to maximize coherence because the outputs predicted by $N_{Coh}(i)$ at the algorithmic level are different from the outputs predicted by $\psi_{Coh}(i)$ at the computational level, leading to an internal inconsistency in the theory composed of both explanations (see Thagard, 2000).

One may contend that N_{Coh} may be a veridical algorithmic-level description for the class of inputs relevant for cognizers, and possibly this is what Thagard had in mind. But this just raises the question of what those inputs are. Let us denote this special set of inputs by $I' \subseteq I$. Then, the claim is that N_{Coh} is an exact algorithm (not a heuristic) for the function $\psi'_{Coh} : I' \rightarrow O$. But that just means that, assuming that N_{Coh} is veridical at the algorithmic level, that ψ'_{Coh} (but *not* ψ_{Coh}) is the veridical model at the computational level. Besides, if N_{Coh} is indeed veridical at the algorithmic level, then on the Tractable Cognition thesis, ψ'_{Coh} is a tractable function, although ψ_{Coh} is not. Rejecting ψ_{Coh}, and replacing it by ψ'_{Coh}, is exactly what the *Theory revision* reaction requires, and this is what Thagard could have done to maintain internal consistency of his explanation of coherence. Doing so, however, does require the identification of I' as well as some argument making plausible that the class of inputs for which people can compute maximum coherence is satisfactorily modeled by I'.

Now, approximation algorithms may not seem to be susceptible to the same criticism as heuristics because an approximation algorithm—with its provable performance bound—does have a well-defined relation to the function it is approximating. Still, some inconsistency exists between an "exact" computational-level theory and an "approximate" algorithmic-level description. Some may argue that certain "small" degrees of divergence between computational-level predictions and algorithmic-level predictions are acceptable, provided the divergence is

contained and well-understood (as is the case of approximation algorithms proper). Be that as it may, I think nothing is lost, and both tractability and consistency are won, if the relevant form of approximation were simply made part of the computational-level theory. To illustrate, let A be an algorithm that tractably approximates some intractable function ψ_T. Then, instead of maintaining the intractable ψ_T as computational-level theory of some cognitive capacity ψ (according to the Tractable Cognition thesis, ψ_T must be a non-veridical theory of ψ anyway), we may formulate a new computational-level theory $\psi_{T_{approx}}$, which corresponds to the approximation problem computed by A. This approximation problem is tractable and hence may, for all we know, be a veridical computational-level theory of the cognitive capacity of interest. Clearly, this is just another way to implement a *Theory revision* response. Note also that the approximation algorithm A for ψ_T will now automatically be an *exact* algorithm for $\psi_{T_{approx}}$, which ensures consistency between the explanations at the algorithmic level (A) and the computational level ($\psi_{T_{approx}}$).

One is warned that, for purposes of cognitive modeling, not just any index of approximation will do. To illustrate consider again the neural network processor, N_{Coh}. In their Appendix, Thagard and Verbeurgt (1998) proved that, given certain amount of preprocessing, N_{Coh} can be made to approximate ψ_{Coh} in the sense that it computes a truth assignment whose coherence level is within 87% of the maximum coherence level (e.g., if the maximum coherence level is 100, then the algorithm always outputs a truth assignment with coherence at least 87). As Millgram (2000) rightly pointed out, however, approximating the maximum coherence value is not the same as approximating the truth assignment having the maximum coherence value. In this particular case, the two can differ by an arbitrarily large amount. Namely, a truth assignment with almost maximum coherence can be exactly the opposite of the maximum coherence truth assignment itself. The potential misfit between approximating optimal values and approximating structural aspects of optimal outputs seems to have been a largely ignored phenomenon in both the computer science and the cognitive science literature. The topic deserves further investigation, especially because more "approximation theories" will likely emerge the more often the Tractable Cognition thesis is put into practice in cognitive science (see Hamilton, Müller, van Rooij, & Wareham, 2007, for a first study along these lines).

Last, I note that my criticisms of the *Framework rejection* and *Devising heuristics* responses to intractability are not specific to the use of the P-Cognition thesis. They would apply with the same force had the researchers adopted a different formalization of the Tractable Cognition thesis. It just so happens that the only accepted formalization to date has been the P-Cognition thesis. In the next section I put forth a different formalization in the form of the FPT-Cognition thesis. Like the P-Cognition thesis, the FPT-Cognition thesis supports only the *Theory revision* response to an NP-hardness result, but it does allow for a wider range of adjustments.

5. The FPT-Cognition thesis

The FPT-Cognition thesis builds on the relatively young theory of parameterized complexity, founded by Downey and Fellows (1999; see also Fellows, 2002; Flum & Grohe, 2006;

Niedermeier, 2006). Parameterized complexity theory is motivated by the observation that some NP-hard functions can be computed by algorithms whose running time is polynomial in the overall input size $|i|$ and non-polynomial only in some small aspect of the input called the input parameter. In other words, the main part of the input contributes to the overall complexity in a "good" way, whereas only the input parameter contributes to the overall complexity in a "bad" way. In these cases, we say that the parameter confines the non-polynomial time complexity inherent in ψ_T, and the function is said to be fixed-parameter tractable for that parameter. More formally, a function ψ_T is *fixed-parameter tractable* for a parameter k if there exists at least one algorithm that computes ψ_T in time $O(f(k)|i|^{\alpha})$, where f is a function depending only on the parameter k and α is a constant. In this case, the algorithm is called a fixed-parameter tractable (fpt-) algorithm and the parameterized function k-ψ_T is said to belong to the class FPT. Parameterized functions that do not belong to FPT are called fixed-parameter intractable.

To illustrate, let us consider a weighted graph. This is a commonly assumed input structure for both connectionist and symbolic theories of cognition. The class of weighted graphs on which a hypothesized cognitive capacity operates may be limited in many possible ways. For example, the class may contain only graphs in which any given node has at most k_1 connections to other nodes, or graphs with at most k_2 different connection weights, or graphs in which the distance between any two nodes is at most k_3, or graphs with at most k_4 layers of nodes, or graphs with layers such that nodes have at most k_5 incoming connections and at most k_6 outgoing connections, and so forth. Now, note that even if some function ψ_G defined over input graphs is not computable in polynomial-time $O(|i|^{\alpha})$, it may still be computable in fpt time $O(f(k)|i|^{\alpha})$, for one or more such graph parameters $k \in \{k_1, k_2, k_3, \ldots\}$. In those cases, as long as k is relatively small (e.g., much smaller than the size of the entire graph), the computation of ψ_G may then still be performed quite fast even for large input graphs. This is illustrated in Table 3.

The illustration in Table 3 assumes that k is relatively small (in this case $k = 10$) and, again, processing speed is assumed to be 10,000 computational steps per second. First of all, notice that the fpt running times $O(2^k|i|)$ and $O(2^k + |i|)$ grow much slower with $|i|$ than the exponential running time $O(2^{|i|})$. Hence, in contrast to what the P-Cognition thesis claims, a theory ψ_T that is not polynomial-time computable may still be psychologically feasible, even for large $|i|$, provided only that ψ_T is computable in fpt-time for a parameter that is relatively small in practice. Notice that the requirement of fpt-time is not a luxury, not even for small parameters. As can be seen in Table 3, a running time that is not fpt-time, such as $O(|i|^k)$, is generally impractical even if k is relatively small. Hence, the existence of small input parameters is by itself not sufficient for efficient computability of a function.

These observations lead me to propose the FPT-Cognition thesis: Cognitive functions are among the functions that are fixed-parameter tractable for one or more input parameters that are small in practice. Fig. 7 illustrates how the FPT-Cognition thesis instantiates a relaxation of the P-Cognition thesis (i.e., functions that are classified as intractable under the P-Cognition thesis may still be tractable under the FPT-Cognition thesis).

Why relax the P-Cognition thesis? If we use computational complexity theory to constrain psychological theorizing, we do not want to constrain it too much—that is, we do not want to risk rejecting veridical theories simply because our formalization of the Tractable Cognition

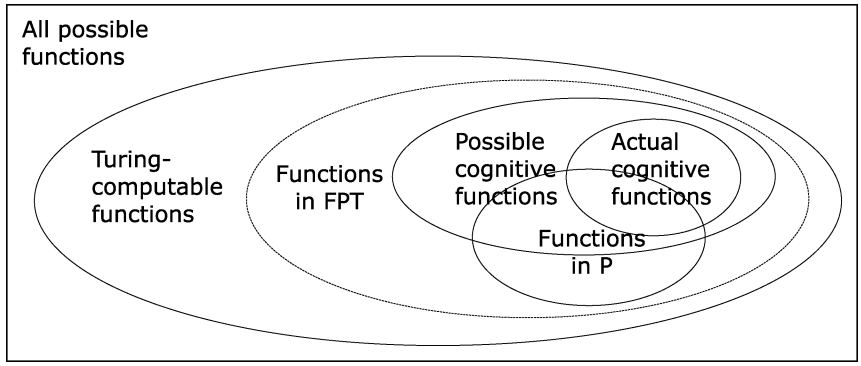

Fig. 7. According to the FPT-Cognition thesis, the set of functions describing *possible* cognitive capacities is a subset of the set of functions that are *fixed-parameter tractable* (FPT) for small parameters. The set FPT includes the functions in P as special cases.

thesis is wrong. Because fpt-algorithms can be feasible despite their super-polynomial complexity, as I have explained above, then if we adopt the P-Cognition thesis we risk exactly this.

Relaxing the P-Cognition thesis as is done by the FPT-Cognition thesis is only useful, of course, if there exist computational-level theories that are not in P but that are in FPT for relevant input parameters. As it turns out, such theories indeed exist. Many apparently intractable computational-level theories are fixed-parameter tractable for one or more parameters that characterize their respective inputs (see, e.g., van Rooij & Wareham, 2008, for a review). We will consider the Coherence theory ψ_{Coh} of Thagard (2000; Thagard & Verbeurgt, 1998) as an illustration:

> Coherence (ψ_{Coh})
> *Input:* A set of propositions $P = \{p_1, p_2, \ldots, p_n\}$ and a set of constraints $C \subseteq P \times P$. Constraints come in two kinds: positive constraints C^+ and negative constraints C^- with $C^- \cap C^+ = \emptyset$.
> *Output:* A truth assignment to the propositions in P that maximizes the number of satisfied constraints in C. Here, a positive constraint $(p_i, p_j) \in C^+$ is satisfied if p_i and p_j are both set to "true" or both set to "false" and a negative constraint $(p_i, p_j) \in C^-$ is satisfied if p_i is set to "true" and p_j is set to "false."

The Coherence theory has several explicit and implicit input parameters. Examples of explicit parameters are the number of positive constraints ($|C^+|$) and the number of negative constraints ($|C^-|$); examples of implicit parameters are the maximum number of constraints that are satisfied (s) or unsatisfied (u) by the optimal assignment. Observe that large belief networks—that is, networks consisting of many connected propositions—may still have small associated $|C^+|$ or small associated $|C^-|$ (although both cannot be small at the same time). Also, large belief networks may still have small associated s or u (although, again, both cannot be small because $u + s = |C^+| + |C^-| = |C|$). Whether or not these parameters are small for real-life human belief networks is an empirical question, but it is conceivable that at least one of them

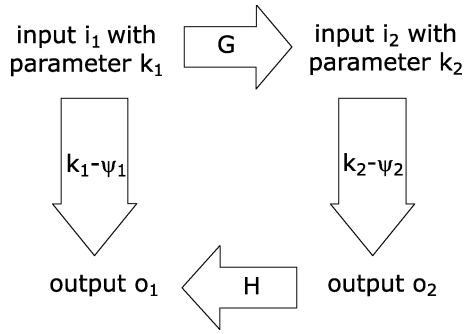

Fig. 8. Illustration of the concept of parametric reduction. The functions G and H together constitute a parametric reduction from k_1-ψ_1 to k_2-ψ_2 if, and only if, $\psi_1(i_1) = H(\psi_2(G(i_1)))$ such that $k_2 = f(k_1)$ and both functions G and H are computable in fpt time. In this case, ψ_1 is fixed-parameter tractable for parameter k_1 if ψ_2 is fixed-parameter tractable for parameter k_2.

is. If the function ψ_{Coh} is fixed-parameter tractable for one or more of these parameters, then despite being NP-hard, the function may still be tractably computable in practice. To investigate this possibility, we need a way to distinguish functions that are in FPT from those that are not. As we will now see, the theory of parameterized complexity provides the necessary tools.

Analogous to the classical method of polynomial-time reduction introduced in section 3, parameterized complexity theory introduces a method called *parametric reduction* (see Fig. 8 for an illustration). A parametric reduction from a parameterized function k_1-ψ_1 to a parameterized function k_2-ψ_2 is a reduction as defined in section 3 with the additional requirements that (a) the transformation from input i_1 for k_1-ψ_1 to input i_2 for k_2-ψ_2 is such that the size of k_2 is bounded by some function $g(k_1)$ and (b) the transformation is computable in fpt-time $O(f(k_1)|i_1|^{\alpha})$. We can use the method of parametric reduction to prove a function k_1-ψ_1 to be in FPT as follows: Take a parameterized function k_2-ψ_2 that is known to be in FPT and devise a parametric function from k_1-ψ_1 to k_2-ψ_2. This suffices to prove k_1-$\psi_1 \in$ FPT because k_1-ψ_1 can then be computed in fpt-time by first transforming its input into an input for k_2-ψ_2, next computing the output of k_2-ψ_2 and transforming that output into the corresponding output for k_1-ψ_1. Using this method it can be shown that $|C^-|$-ψ_{Coh}, s-ψ_{Coh}, and u-ψ_{Coh} are all fixed-parameter tractable.[8] This means that computing function ψ_{Coh} is computationally tractable for all networks with high or low level of coherence and all networks with relatively few negative constraints.[9]

Establishing fixed-parameter intractability is a bit more tricky and often only possible relative to some conjecture, like P $\neq$ NP, which is believed to be true but, so far, unproven. Analogous to the classical complexity class NP that encloses P, parameterized complexity theory introduces a complexity class W[1] that encloses FPT (see Downey & Fellows, 1999, for a definition of the W[1] class). It is widely believed that there exist parameterized problems in W[1] that are fixed-parameter intractable and thus that FPT $\neq$ W[1] (see also Fig. 9). This conjecture is, among other things, motivated by the observation that there exist W[1]-hard functions. A parameterized function k-ψ is W[1]-hard if any parameterized function k'-ψ' in W[1] can be transformed to k-ψ via a parametric reduction (functions that are W[1]-hard and

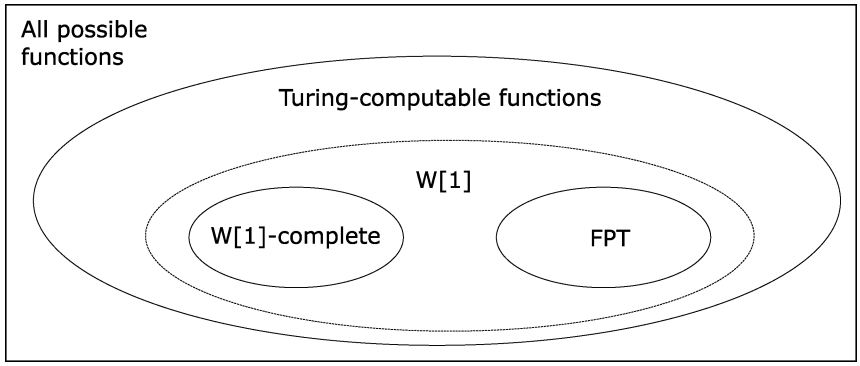

Fig. 9. The view of W[1] on the assumption that FPT $\neq$ W[1]. W[1]-complete parameterized problems are not in FPT if, and only if, FPT $\neq$ W[1].

in W[1] are called W[1]-complete). Note that membership of any W[1]-hard problem in FPT would imply that FPT = W[1]. Therefore, assuming FPT $\neq$ W[1], proving that a function ψ_T is W[1]-hard is proving that $\psi_T \notin$ FPT. In the remainder of this article, we will work under the assumption that FPT $\neq$ W[1].

Proving W[1]-hardness is facilitated by the fact that hundreds of parameterized functions are already known to be W[1]-hard (see, e.g., Downey & Fellows, 1999, and the online compendium of *Parameterized Complexity Results*); a parametric reduction from any one of those functions to k-ψ_T suffices to prove the latter W[1]-hard as well. Using the method of parameterized reduction, we can prove, for example, that $|C^+|$-ψ_{Coh} is W[1]-hard. This means ψ_{Coh} is not fixed-parameter tractable for the parameter $|C^+|$, which in turn means that limiting the input domain to constraint networks with relatively few positive constraints does nothing substantially to make the computation of ψ_{Coh} easier.

Before closing this section, I would like to draw the reader's attention to the nature of the relationship between the classes P, NP, FPT, and W[1]. In relating these classes, it is important to keep in mind that W[1] and FPT are classes of parameterized functions, whereas NP and P are classes of (non-parameterized) functions. The relationship between W[1] and NP can be understood as follows: Let PAR(NP) denote the set of all possible parameterizations of functions in NP, and let PAR(P) denote the class of all possible parameterizations of functions in P; then, PAR(P) $\subseteq$ PAR(NP) and PAR(P) $\subseteq$ FPT $\subseteq$ W[1]. Fig. 10 illustrates these relationships.

5.1. Putting the FPT-Cognition thesis into Practice

Having argued that the FPT-Cognition thesis is a better (less restrictive) formalization of the Tractable Cognition thesis, we now consider how the FPT-Cognition thesis might be usefully employed in psychological practice.

Let me start by emphasizing that by replacing the P-Cognition thesis with the FPT-Cognition thesis I do not mean to argue that NP-hardness results are of no significance for cognitive science. On the contrary, the FPT-Cognition thesis, like the P-Cognition thesis, recognizes that an NP-hard function $\psi_T : I \rightarrow O$ cannot be practically computed in all its generality

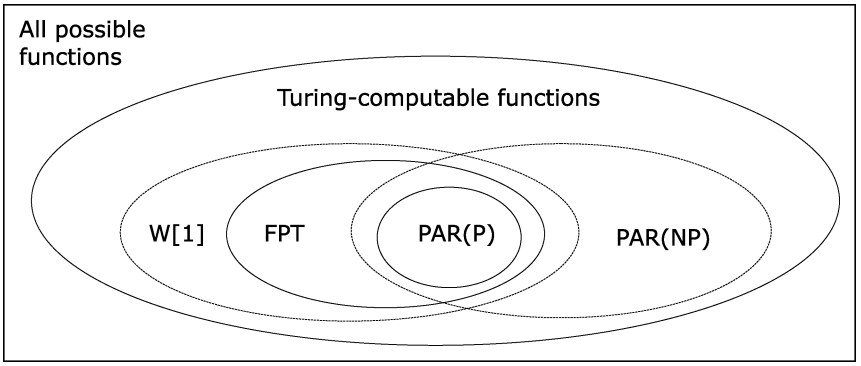

Fig. 10. Illustration of the relationship between classes W[1], FPT, P, and NP. Because W[1] and FPT are classes of parameterized problems we cannot compare them directly to the classes P and NP. Therefore, we define the class of all possible parameterizations of problems in P, denoted PAR(P), and the class of all possible parameterizations of problems in NP, denoted PAR(NP). Because P $\subseteq$ NP, we have PAR(P) $\subseteq$ PAR(NP). Further, because every problem in P is in FPT for any parameter, we have PAR(P) $\subseteq$ FPT $\subseteq$ W[1].

and requires a *Theory revision* response as discussed in section 4, that is, if a cognitive system is computing ψ_T at all, then it must be computing some restricted version $\psi_{T'} : I' \rightarrow O$ where $I' \subset I$; otherwise the cognitive system must be computing some other function $\psi_{T'}$, where $\psi_{T'}(i) \neq \psi_T(i)$ for some $i \in I$. The difference lies in what the two theses consider to be feasible hypotheses for $\psi_{T'}$. According to the P-Cognition thesis, $\psi_{T'}$ must be polynomial-time computable, whereas the FPT-cognition thesis only requires that $\psi_{T'}$ is fixed-parameter tractable for one or more input parameters that are small in practice.

Let me also comment on the qualification "small in practice" in the definition of the FPT-Cognition thesis. This qualification is motivated by the observation that speed of an fpt-algorithm depends both on (a) the exact form of the function $f(k)$ in the running time and (b) the range of values that k may take in practice. For example, the range of feasibility for k is different for an fpt-algorithm that runs in $O(2^k|i|)$ than for an fpt-algorithm that runs in, say, $O(k!|i|)$. The formal theory of parameterized complexity can help determine bounds on the function $f(k)$, but estimates of bounds on the parameter k itself must be informed by psychological theory and empirical observation. Although the qualification may seem to make the FPT-Cognition less formal than the P-Cognition, one is reminded that implicitly the P-Cognition is also assuming that the constants in the polynomials are small; clearly, a polynomial running time of $O(|i|^{\alpha})$ is unfeasible if, say, $\alpha \geq 10$ (see, e.g., column 3 in Table 3). More important, one must keep in mind that formalism should not come at the expense of realism and applicability. The informal requirement that parameters be relatively small is exactly what brings the FPT-Cognition thesis much closer to psychological reality than the P-Cognition thesis. Many natural cognitive functions have input parameters that are of qualitatively different sizes. Ignoring these qualitative differences, and treating the input always as one big "chunk," makes complexity analysis completely vacuous for purposes of psychological science. The FPT-Cognition thesis recognizes that different input parameters differentially contribute to a function's computational complexity and encourages the cognitive scientist to systematically study this interaction between function and parameters.

Table 3
Illustration of how the running times $O(2^{|i|})$, $O(|i|^k)$, $O(2^k|i|)$, and $O(2^k + |i|)$ compare for different levels of $|i|$ and k fixed at 10 (assuming 10,000 computational steps per second)

$\|i\|$	$O(2^{\|i\|})$	$O(\|i\|^k)$	$O(2^k\|i\|)$	$O(2^k + \|i\|)$
2	0.02 msec	0.10 sec	0.20 sec	0.10 sec
5	0.19 msec	16.3 min	0.51 sec	0.10 sec
10	0.10 sec	11.6 days	1.02 sec	0.10 sec
15	3.28 sec	22 months	1.54 sec	0.10 sec
20	1.75 min	32.5 yrs	2.05 sec	0.10 sec
30	1.2 days	19 centuries	3.07 sec	0.10 sec
50	35 centuries	3.1×10^5 years	5.12 sec	0.10 sec
100	4.0×10^{18} years	3.2×10^8 years	10.2 sec	0.10 sec
1,000	3.4×10^{290} years	3.2×10^{18} years	1.71 min	0.20 sec

The FPT-Cognition thesis, then, should not be seen as a simple litmus test for distinguishing feasible from unfeasible computational-level theories. On the contrary, the FPT-Cognition thesis, as I propose it here, is meant to stimulate active exploration of natural input parameters of cognitive capacities and to systematically study how they contribute to the computational complexity of computational-level theories. It is only when we know how the complexity of a function depends on its problem parameters that we can have a solid understanding of its computational complexity in practice. Ideally, a cognitive scientist confronted with the intractability of his or her computational-level theory would proceed as follows:

1. Make explicit assumptions about (bounds on) the range of values that relevant input parameters can take in reality.
2. Analyze the parameterized complexity of the theory for all relevant input parameters that can reasonably be assumed to be much smaller than the whole input size.
3. Derive predictions from function restrictions to *tractable* parameter ranges and put them to an empirical test.
4. If no tractable parameter ranges can be found, or none that passes empirical tests, then revise the theory and return to Step 1.

Because the tractability requirement in Step 3 of this cycle can force theory predictions that otherwise could not be derived, implementing the FPT Cognition thesis in modeling practice, as suggested here, helps psychologists perform more rigorous tests of their computational-level theories, and, as such, it helps reduce the degree of the empirical underdetermination of computational-level theories.

6. Possible objections

It is my experience that the ideas expressed in this article often give rise to questions or criticisms by cognitive scientists and psychologists. This section discusses 11 common

objections that I encountered in discussions with colleagues. By facing these objections head-on, I hope to clarify what the Tractable Cognition thesis can do for us and what it cannot do. Furthermore, I hope to convince the skeptics that the concerns that they raise about the validity of Tractable Cognition thesis, or its proposed formalization, are often based on misconceptions or uninformed intuitions. Each of the objections that I discuss in this section arises from a particular theoretical perspective. I distinguish between three perspectives: the perspective of (A) a researcher who subscribes to the P-Cognition thesis, but who questions the FPT-Cognition thesis; (B) a researcher who subscribes to the computational approach to cognition, but who questions the Tractable Cognition thesis; and (C) a researcher who does not subscribe to the computational approach to cognition. I indicate for each objection the perspective from which I believe it arises.

6.1. The empiricist objection

> Perspective B or C: *Cognitive models should be evaluated on how well they explain empirical data, not on their* a priori *plausibility.*

This objection is often raised by experimental psychologists, who may feel that the Tractable Cognition thesis is a rationalist attempt to accept or reject cognitive models solely on theoretical grounds, a practice that would seem to contradict all that experimental psychology stands for. But the Tractable Cognition thesis is not intended to replace empirical evaluation of cognitive models at all. It is well understood that, in the end, all cognitive models must stand the test of empirical scrutiny. To optimally benefit from both tractability and empirical constraints on cognitive models, I have even argued that tractability testing and empirical testing should be interlaced (see section 5.1). What the Tractable Cognition thesis does offer is a way of recognizing computationally unfeasible cognitive models or model variants that can be rejected even before they need to be put to an empirical test. Even if such a computationally unfeasible model were to fit the empirical data, it would not be able to *explain* the data because explanation requires more than fit alone (Cummins, 2000). For example, the model should make insightful how the cognitive capacity could generate the observed data, which a computationally unrealistic model cannot. In sum, the Tractable Cognition thesis helps constrain the vast space of possible cognitive models one may postulate for any given cognitive capacity. This seems particularly helpful because many such models are about unobservable, or only indirectly observable, cognitive processes; and thus, they are typically vastly underconstrained by the available observational data (see also Anderson, 1987, 1990).

If one is a non-computationalist (Perspective C), then one may not recognize the tractability constraint on cognitive models. In that case, refer to my response to the *cognition-is-not-computation objection* below.

6.2. The cognition-is-not-computation objection

> Perspective C: *Computational complexity theory has nothing to offer cognitive science because cognition is not computation.*

What is meant by this objection depends crucially on the meaning of the phrase "cognition is not computation." I believe the phrase is associated with a multitude of meanings. In my reactions below, I distinguish between four possible versions of the objection:

> Version 1: *Complexity analysis does not apply to cognition because cognition is not symbolic computation.*

In cognitive science, the terms *computation* and *computationalism* have become associated with the symbolic tradition, and sometimes even with particular models in this tradition (e.g., Anderson, 1987; Fodor, 1987; Newell & Simon, 1988a, 1988b; Pylyshyn, 1984, 1991). All that the Tractable Cognition thesis requires, however, is a commitment to a *minimal computationalism* (Chalmers, 1994). Many theories that are considered "non-computational" may still fall under the heading of computationalism in this broad sense. For example, despite their presumed non-computational status (e.g., Port & van Gelder, 1995; Thelen & Smith, 1994; van Gelder, 1995, 1998, 1999), dynamical systems models can be subjected to computational complexity analysis (Siegelmann & Sontag, 1994).

> Version 2: *Complexity theory does not apply to cognitive systems, because cognitive systems do not realize input/output functions.*

In this article we assumed that cognitive capacities can be understood (at Marr's computational level) as the computation of input/output functions. It may be, however, that the purpose of some (or all) cognitive systems is not to compute any input/output functions at all. Instead, their purpose may be to cycle through a set of states indefinitely, without ever halting and producing an output (cf. Levesque, 1988, p. 385). Such processes may seem to fall outside the scope of the Tractable Cognition thesis. However, this is not necessarily the case. Even unhalting or cyclical processes can be conceptualized as computations of input/output functions where the inputs and outputs are internal states. Consider, for example, those processes whose functionality is to maintain certain (representational) states over time or to ensure that certain system values do not grow out of bounds (e.g., homeostatic processes). Such so-called "maintenance" computations are similarly constrained by the requirement of tractability and models about them can be subjected to the same types of computational complexity analyses as standard input/output computations (see, e.g, Dunne, Laurence, & Wooldridge, 2003; Wooldridge & Dunne, 2005).

> Version 3: *Cognitive functions need not be computationally tractable because cognitive systems realize their input/output mappings via non-computational means.*

Some non-computationalists do not question that cognitive capacities realize input/output mappings (i.e., they do not subscribe to Version 2), but they propose that cognitive systems realize such mappings in non-computational ways (e.g., Horgan & Tienson, 1996; Krueger & Tsav, 1990). On this view, cognitive functions need not be computationally tractable. Be that as it may, the non-computationalist is faced with the problem that non-computationalist explanations seem to be made of mysterious stuff and lack explanatory value (see also Cherniak, 1986; Levesque, 1988). John Tsotsos (1990) put it as follows:

> Experimental scientists attempt to explain their data, not just describe it .... There is no appeal to non-determinism or to oracles that guess the right answer or to undefined, unjustified, or "undreamed of" mechanisms that solve difficult components. Can you imagine theories that do have these characteristics passing a peer-review procedure? (p. 466)

In a similar vein, Prasse and Rittgen (1998) objected to Wegner's (1997) claim that "interactive machines" would be able to exceed the computing powers of Turing machines by performing "non-algorithmic procedures." Prasse and Rittgen explained how the purported non-algorithmic aspects of interactive machines seem to be an artifact of Wegner leaving the process of interaction unmodeled, undefined, and hence unexplained. They furthermore illustrated that existing well-defined models of interactive computability are known not to supersede Turing computability (see also Cockshott & Michaelson, 2005; Wegner & Eberbach, 2004):

> Version 4: *Complexity theory does not apply to cognition because computation is an altogether wrong way of thinking about cognition.*

This last version of the *cognition-is-not-computation objection* represents the non-computationalist that is not persuaded by any of my reactions to Versions 1 through 3. In my opinion, even this researcher should appreciate the contribution that the Tractable Cognition thesis makes to cognitive science; namely, a non-computationalist can still recognize that tractability is a constraint on computational theories of cognition. Then, the Tractable Cognition thesis offers the non-computationalist a way of evaluating the success of his or her competition. If, in the long run, human cognition systematically defies tractable computational description, then this can be taken as empirical support for the idea that computation is the wrong way of thinking about cognition.

6.3. The super-human objection

> Perspectives B or C: *Humans are known to perform tasks that are computationally intractable. This goes to show that tractability is not a constraint on human computation.*

Some cognitive tasks that are performed effortlessly by humans are presently being modelled by computationally intractable functions (e.g., Haselager, 1997; Oaksford & Chater, 1993, 1998; see also Table 2). Some researchers interpret this as evidence that people can realize computationally intractable capacities (e.g., Siegel, 1990). In my view, the argument is flawed. There are two possibilities: Either one is a computationalist (Perspective B), or one is not (Perspective C). If one is, then one should concede that either the capacities are incorrectly modelled (i.e., requiring a *Theory revision* response as discussed in section 4) or that the wrong criterion for tractability has been adopted. If one is not a computationalist, then one does not recognize that the models truly capture the nature of the capacity in the first place, and thus their classification as "intractable" is irrelevant to the conceptualization of the capacity (unless Version 3 of the *cognition-is-not-computation objection* applies. In that case see my response to that objection above).

6.4. The heuristics objection

> Perspective B: *Humans often use heuristics instead of exact algorithms. Then intractability is not an issue.*

In my response to this objection, I distinguish between two possible meanings of the word *heuristic*: one pertaining to the computational level of explanation (heuristic$_1$) and one pertaining to the algorithmic level (heuristic$_2$). I will argue that the objection is unfounded for both meanings.

An input/output function postulated at Marr's computational level may be a heuristic$_1$ procedure for solving a real-world problem (assumed to be) faced by the cognitive system. This heuristic$_1$ does not guarantee that the problem is always solved in all situations, but it is believed to solve the problem often enough or close enough to be of practical use. It is a mistake to think that such a heuristic$_1$ is always computationally cheap or even tractable. Take, for example, the problem faced by people to decide what to believe and what not. There does not seem to exist any procedure that given observations about the world produces only true (non-trivial) beliefs. Coherentists propose that a good heuristic$_1$ for coming to have many true beliefs is to maximize the coherence among one's beliefs and observations. But note that, despite its heuristic$_1$ status, the computational-level model of Thagard for coherence maximization, ψ_{Coh}, is known to be computationally intractable (i.e., NP-hard). According to the Tractable Cognition thesis, we can thus reject the heuristic$_1$ ψ_{Coh} as a descriptive theory of human coherence maximization.

One may object that a heuristic$_1$ being computationally intractable does nothing to affect its descriptive validity at the computational level because humans may "approximate" heuristic$_1$ using a heuristic$_2$ (i.e., an inexact procedure postulated at the algorithmic level). This approach is indeed the one pursued by Thagard upon finding that ψ_{Coh} is intractable. Because I already explained in section 4 why this approach is explanatorily incoherent (see also van Rooij & Wright, 2006), I mainly restate the conclusions of that discussion here: A heuristic$_2$ H for ψ_{Coh} does not compute ψ_{Coh} but rather some other function ψ'_{Coh} (which may be superficially similar to ψ_{Coh}, but still importantly different, because one can be intractable while the other is not). Therefore, it is ψ'_{Coh}, and not ψ_{Coh}, that should be adopted as the computational-level model if H is adopted as algorithmic-level theory. Note that H hereby ceases to be a heuristic$_2$ and instead becomes an exact algorithmic level explanation of how the system computes heuristic$_1$ ψ'_{Coh}, as it should be. Now observe that if H is tractably computable, then so is ψ'_{Coh}.

6.5. The average-case objection

> Perspective B: *A computation that is classified as intractable on a worst-case analysis may still be tractable in practice. An average-case measure of complexity should be used instead.*

In this article we have adopted a worst-case measure of complexity, in the sense that the complexity of an input/output mapping is defined by its hardest input. This measure is appropriate for assessing (in)tractability of computational-level models because for a computation

to be tractable it must be tractable for *all* its possible inputs, including the worst-case input. One possible objection may be that the worst-case input may never happen in practice and that, therefore, the worst-case complexity measure overestimates the real complexity of the computational-level theory. This objection becomes self-contradictory, however, once we accept that the hypothesized input domain is part of the computational-level theory (as I have done throughout this article): The cognitive modeler postulating a computational-level theory $\psi_T : I \rightarrow O$ is, by definition, assuming that every input $i \in I$ can happen in practice. If there were to exist any input $i \in I$ for which the modeler believed it could not or does not happen in practice, then he or she should instead postulate a different computational-level theory $\psi_{T'} : I' \rightarrow O$, where input domain $I' \subset I$ is restricted so as to exclude those inputs not assumed to occur. The new theory $\psi_{T'}$ again has its own worst-case inputs. Clearly, a modeler could mistakenly be assuming that an input situation i arises in the real world when in fact it never does. Then a worst-case analysis of the computational-level model could indeed overestimate the worst-case complexity of the computation performed by the relevant cognitive capacity in the real world. But this mismatch would be due to a mistake on the side of the modeler in hypothesizing the capacity's input domain, *not* a reason to object to worst-case analysis. After all, both the computational-level theory and the real-world capacity have a worst-case scenario, and the problem here is that the two worst-case scenarios do not match up, *not* that an average-case analysis of the computational-level theory would have been more appropriate.

This is not to say that average-case complexity is never a preferred measure of complexity. It may very well be for purposes other than assessing tractability, for example, when a cognitive psychologist is interested in comparing the time-complexity of different (tractable) algorithmic-level explanations with reaction time data obtained via experimentation. In this case, comparing mean reaction time with the algorithms' average-case complexity may be more informative than comparing it with their worst-case complexity (cf. Dry, Lee, Vickers, & Hughes, 2006; Pizlo et al., 2006).

6.6. The parallelism objection

> Perspective B: *Cognitive computation is (to a large extent) parallel, not serial. A function that is intractable for a serial machine need not be intractable for a parallel machine.*

The complexity measures adopted in this article are defined with reference to computation by serial Turing machines. Nevertheless, the arguments put forth here for the Tractable Cognition thesis can naturally be extended to include parallel computation (see also Frixione, 2001). From a complexity perspective, the difference between serial and parallel computation may be quite insubstantial as far as tractability is concerned. To illustrate, let us first consider a parallel machine M with c processing channels, such that M computes a given serial computation by performing c steps in parallel (i.e., simultaneously).[10] Further, let ψ_T be an input/output function with time-complexity $O(f(n))$, where $n = |i|$ is a measure of input size. Then M computes ψ_T at best in time $O(\frac{f(n)}{c})$. Note that, if c is a constant, then the speed-up due to parallelization is by a constant factor only, and $O(\frac{f(n)}{c}) = O(f(n))$. The speed-up factor c

can be taken into account in the analysis—there is nothing inherent in complexity theory that prevents one from doing so—but if $f(n)$ is a non-polynomial function, then the speed-up due to c becomes negligibly small very fast as input size n grows (see also Table 1). This means that non-polynomial-time complexity computations are unrealistic also for parallel machines, for all but small inputs. The same is true if c is bounded by some polynomial function of n. If c were to grow non-polynomially as a function of n, then $O(\frac{2^n}{c}) = O(f(n))$ may be a polynomial running-time. In that case, indeed time would no longer be a limiting factor, but the space required for implementing the astronomical number of parallel processing channels would be.

In other models of parallel computation, the speed-up due to a constant c need not be constant, but may grow with n. This is the case, for example, in the parallel random access machine model, where a parallel machine M is assumed to have available c processors that can all communicate to each other in constant time (e.g., Gibbons & Rytter, 1988). Although such a parallel machine can compute certain computations faster than a serial Turing machine, the difference in speed between the two machine models is never more than a polynomial amount of time, and, hence, negligible for non-polynomial time computations (see, e.g., Jaja, 1992; van Emde Boas, 1990). The same is true for other reasonable models of parallel computation (e.g., circuits, Parberry, 1994; neural nets, Siegelmann & Sontag, 1994). In general, it is believed that for any reasonable parallel machine the speed-up due to c will be by at most a polynomial amount and, hence, the Invariance thesis applies both to serial and parallel computation (Frixione, 2001; Parberry, 1994; Tsotsos, 1990). This means that if the Invariance thesis is true, then parallel machines cannot compute functions outside P in polynomial-time nor compute parameterized functions outside FPT in fpt-time.

6.7. The non-determinism objection

> Perspective B or C: Cognitive systems are not deterministic machines, and thus functions that are intractable for deterministic machines may still be tractable for cognitive systems.

The complexity measures that we adopted in this article are defined with reference to computation by deterministic Turing machines. If indeed P $\neq$ NP, then we know that there exist functions that cannot be computed in polynomial-time by any deterministic Turing machine, but that can be computed in polynomial-time by a non-deterministic Turing machine. This shows that non-deterministic computations can be more powerful than deterministic computations, at least as far as polynomial-time computability is concerned.[11] This observation has been misinterpreted by some researchers as showing that probabilistic or randomized computation is more powerful than deterministic computation (see e.g., Martignon & Hoffrage, 2002). This is an invalid inference, however, presumably based on an equivocation on the word non-determinism. Below I clarify the important difference between *non-determinism* as it is understood in computation theory and non-determinism as it applies to probabilistic models of mind.

In computation theory, a non-deterministic Turing machine is one that can pursue an unbounded number of possible computation paths in parallel (see the Appendix). Such a

non-deterministic machine M "computes" a function $\psi : I \rightarrow O$ if for all i, at least one of the possible paths leads to an output corresponding to $\psi(i)$. In other words, M can be seen as a serial "oracle" that always knows (magically) which guess to make at every fork in the road. This conceptualization makes clear why non-deterministic Turing machines are generally not considered reasonable models of resource-bounded computation. Consider a probabilistic interpretation of M, called M', which has exactly the same set of possible computation paths as M; but in contrast to M, machine M' can make random guesses about which computation paths to pursue. Such a probabilistic machine M' is said to "compute" a function $\psi : I \rightarrow O$ *with high probability* if for all i the probability that the output of M' corresponds to $\psi(i)$ is large (e.g., approaches 1). Now note that if M has an arbitrarily large number of possible computation paths, corresponding to an arbitrarily large number of possible outputs for M', then the probability that M' "guesses" the right output for any given input can be arbitrarily small. This means that the non-deterministic computation power of M is not automatically matched by the probabilistic computation power of M'.

This is not to say that probabilistic computational models of cognitive systems are impractical or useless. On the contrary, they may very well provide better models of certain cognitive systems than deterministic models. But one should not be fooled into thinking that probabilistic machines have the same abilities as non-deterministic machines. Even if it turns out that probabilistic machines have different abilities than deterministic machines, this does not obviate complexity analyses in cognitive psychology. Tractability is as much a requirement on probabilistic computation as it is on deterministic computation. Moreover, current theoretical investigations into the computational power of probabilistic computation are generating evidence that NP-hard input/output functions cannot be computed in polynomial-time by probabilistic machines unless P = NP. This provides hope that formalizations of the Tractable Cognition thesis for deterministic cognitive computations can be generalized directly to a formalization of the thesis for probabilistic cognitive computations.

6.8. The quantum computing objection

> Perspective B: *Cognitive systems may have quantum computing powers, and if they do then they can efficiently compute functions that are intractable for classical computing machines.*

This objection is in a sense a combination of the parallelism objection and the non-determinism objection, as a quantum computer is a (theoretical) machine that performs a probabilistic computation using a (potentially unbounded) number of parallel channels. For the formalisms underlying this non-classical model of computation, I refer the reader to Deutsch (1985), Fortnow (2003), and Arora and Barak (in press). The following informal characterization may suffice to support the reader's intuitions about the nature and computational powers of quantum computers: A quantum computer can be thought of as a non-deterministic Turing machine where each computation path exists in a different, parallel universe (or, mathematically equivalently, each path exists only in potentiality in this universe). To decide which of the potentially many paths

is to be selected as the actual computation, a quantum computer can resort to a process called "quantum interference," which is a computational operation that can only expensively be simulated by a deterministic Turing machine by considering all possible computational paths explicitly (for an explanation of the workings of quantum interference, see Arora & Barak, in press).

Let us now analyze the quantum computing objection into its two component parts and assess their respective plausibility. First, how likely is it that cognitive systems have quantum computing powers? At present, there is no evidence that human brains can reliably utilize quantum effects for purposes of cognitive computation, and there is some evidence that they cannot (Grush & Churchland, 1995; Litt et al., 2006). For one, the physical conditions required for stable superimposed states and error-free quantum interference do not seem to be met by the physical milieus that exist in human brains (Litt et al., 2006). Also, as argued by both Litt et al. and Grush and Churchland, there are as of yet no examples of cognitive phenomena that are amenable to quantum computational modeling but not to classical computational modeling. For example, arguments made by Penrose (1994, 1997) that Gödel's first incompleteness theorem would imply that human mathematical understanding requires non-computable (and presumably quantum type) processes in the brain turn out to be fallacious (Grush & Churchland, 1995). Be that as it may, we cannot completely rule out the possibility of quantum computing powers in human brains. So, we may ask, if we grant Part 1 of the objection, to what extent is there substance to Part 2: That is, can quantum computers really compute intractable functions?

As the informal characterization given above may make clear, a quantum computer can potentially draw upon an exponential number of computation channels while using, at most, a polynomial amount of space (in any given universe). In other words, quantum computation can lead to an exponential speed-up, without being subject to the same criticism as classical parallel machines (see my response to the *parallelism objection*). This ability has been misread by some researchers as implying that quantum computers can compute NP-hard problems in polynomial time (Kak, 2000; Narayanan, 1999). This overlooks the computational limits of quantum interference. It is true that there can exist functions outside P that quantum computers may be able to compute in polynomial time. The so-called Integer Factorization problem (Shor, 1997) provides an example: No polynomial-time algorithm is known (nor believed) to exist for Integer Factorization, yet the problem is efficiently computable by a quantum computer. Contrary to claims made by Narayanan (1999), however, Integer Factorization is *not* (known to be) NP-hard. There are an infinite number of complexity classes inbetween P and NP (Garey & Johnson, 1979; of course, unless P = NP), and the Integer Factorization problem belongs to one such class, known as BQP, where it is conjectured that $P \subset BQP \subset NP$ (informally, BQP is the class of polynomial-time quantum computations). There is no NP-hard problem known to be polynomial-time quantum computable, and assuming $BQP \subset NP$, no such NP-hard problem can exist. Moreover, there are few problems outside P known to be amenable to polynomial-time quantum computation. In sum, even if BQP were used as a formalization of the Tractable Cognition thesis, it would instantiate a much more restricted relaxation of the P-Cognition thesis than the FPT-Cognition thesis. After all, many NP-hard problems are known to be fixed-parameter tractable for one or more parameters.

6.9. The small-inputs objection

Perspective B: *For some cognitive capacities, the size of the input is small in practice. In those cases, intractability is not an issue.*

Although the statement is true, it should not be considered an objection to the Tractable Cognition thesis. For example, the statement is perfectly in line with the FPT-Cognition thesis because all input/output functions are in FPT when parameterized by their input size. Thus, if the input size is small then, on the FPT-Cognition thesis, the problem is classified as tractable. It should be noted that also no one who subscribes to the P-Cognition thesis would claim that tractability is an issue if input size is small. The problem is, of course, that for many cognitive capacities the size of the input as a whole is not small (or at least not small enough; see, e.g., Thagard & Verbeurgt, 1998; Tsotsos, 1990). It is then that the P-Cognition thesis and the FPT-Cognition thesis diverge (compare Tables 1 and 3).

6.10. The nothing-new objection

Perspective A: *The requirement that a computational-level theory be in FPT for some "small" input parameters is not essentially different from the requirement that the theory is in P for a restricted input domain.*

The claim reflects a misunderstanding about the relationship between P and FPT. Parameterization is not the same as input restriction; it just determines how we analyze and express the complexity of an input/output function. Recall that parameterized complexity theory expresses a computation's time-complexity as a function of *both* input size $|i|$ *and* a specified parameter k. If a computational-level theory ψ_T, with input parameter k, can be computed by an algorithm that runs in time $O(f(k)n^{\alpha})$, with $n = |i|$, then the *parameterized* function k-ψ_T is said to belong to the class FPT. It is important to realize that in this analysis the value of k is not fixed to be constant—it remains a variable just like $|i|$.[12] It is true that if we were to fix k to be constant, then $O(f(k)n^{\alpha})$ would be $O(n^{\alpha})$, and thus polynomial time, but the same is true also for some fixed-parameter intractable problems (e.g., a running time $O(n^{k})$ is not fpt-time for parameter k but is polynomial-time if k is a constant). This means that the requirement that a computational-level theory be in FPT for some small parameter k is more stringent than the requirement that a computational-level theory is in P for constant k. At the same time, the requirement that the computational-level theory be in FPT for some small parameter k is more lenient than the requirement that the computational-level theory be in P. The FPT-Cognition thesis is thus really a new and importantly different formalization of the Tractable Cognition thesis.

6.11. The too-liberal objection

Perspective A: *Polynomial-time computability is already a too liberal constraint on computational-level theories. Thus, the FPT-Cognition thesis only makes matters worse.*

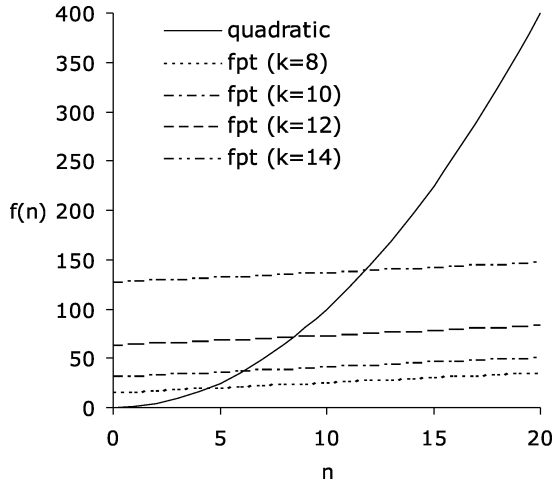

Fig. 11. The figure compares quadratic polynomial-time $O(n^2)$ with fpt-time $O(\sqrt{2}^k + n)$ for parameter $k =$ 8, 10, 12, and 14. *Note*: Increasing the value of k causes the curve for fpt-time to translate upwards.

Some researchers subscribe to the idea that most, if not all, (higher order) cognitive functions are of extremely low complexity (e.g., Gigerenzer & Goldstein, 1996; Martignon & Hoffrage, 2002; Martignon & Schmitt, 1999; Todd & Gigerenzer, 2000). For example, cognitive computations "requiring more than a quadratically growing number of computation steps already appear to be excessively complicated" to Martignon and Schmitt (p. 566). To such researchers, the proposal that cognitive systems perform non-polynomial fpt-time computations may seem outrageous: If the P-Cognition thesis is already too liberal, then its relaxation in the form of the FPT-Cognition thesis may seem to only make matters worse. Two responses to this *too-liberal objection* can be made. First, for purposes of constraining the space of feasible cognitive theories, it is better to use a (slightly) more liberal criterion for formalization than a too strict formalization because the latter will risk excluding veridical theories from consideration. It is for this reason that I have proposed to replace the P-Cognition thesis by the relaxation in the form of the FPT-Cognition thesis. Second, it is not true that fpt computation is necessarily of higher complexity than a polynomial-time computation—it all depends on the size of the parameters and the function describing the algorithm's running time. Fig. 11 illustrates, for example, how an algorithm that runs in non-polynomial fpt time can be much faster than an algorithm that runs in polynomial time, provided only that the parameter k is small.

7. Conclusion

This article can be seen as one of several attempts to establish computational complexity theory as a standard analytic tool in cognitive science for analyzing computational-level theories (see, e.g., Frixione, 2001; Levesque, 1988; Tsotsos, 1990; van Rooij, Stege, & Kadlec, 2005, for other attempts). Large-scale utilization of this tool requires both a recognition of

its value and the means to use it. With this article, I hope to have contributed on both accounts.

I have separated the formulation of the informal Tractable Cognition thesis (that human cognitive capacities are confined to those input/output functions that can be realized using a realistic amount of computational resources) from the question of how the thesis best be formalized. Until now, the informal thesis and its formalization in terms of the P-Cognition thesis have always been conflated, making it hard to assess the arguments for or against one and the other. The conceptual separation made in this article helps open up and clarify such debates. On the one hand, it affords a consideration and discussion of the *in principle* utility of a tractability constraint independent from particular theories of computational complexity. On the other hand, it allows for the utility of different possible formalizations for constraining computational-level cognitive theories to be assessed against the background of an accepted in principle tractability constraint.

The treatment makes clear that the P-Cognition thesis, dominant in current cognitive science, is but one possible formalization of the Tractable Cognition thesis. Moreover, it is arguably not even a good formalization; namely, according to the P-Cognition thesis, all computational-level theories that require exponential-time (or, in general, super-polynomial time) algorithms should be rejected on theoretical grounds. This overlooks the possibility that exponential-time algorithms can run fast, provided only that the super-polynomial complexity inherent in the computation be confined to one or more small input parameters. Given that cognitive input domains are typically characterized by many different input parameters of widely varying ranges, the younger branch of computation theory—called parameterized complexity—can serve cognitive scientists in characterizing the computational resource requirements of different computational-level theories.

Taking these considerations into account, I have proposed to formalize the Tractable Cognition thesis in the form of the FPT-Cognition thesis. The FPT-Cognition thesis recognizes that NP-hard functions cannot be tractably computed for *all* possible input domains (staying true to the spirit and motivation for the P-Cognition thesis), but at the same time acknowledges that certain exponential-time computations can be computationally feasible for many cognitively relevant input domains (relaxing the requirement of polynomial-time computation imposed by the P-Cognition thesis). Because the FPT-Cognition thesis instantiates a relaxation of the P-Cognition thesis, its ability to constrain the space of feasible computational-level theories in cognitive psychology is somewhat weaker than that of the P-Cognition thesis. Be that as it may, the FPT-Cognition thesis is safer to use because it does not risk the exclusion of veridical computational-level theories in the way that the P-Cognition thesis does.

It is very well possible, and arguably desirable, that future theoretical research and considerations will lead to refinements of the FPT-Cognition thesis. One way of improving the constraining power of the FPT-Cognition thesis, for example, is by bringing in assumptions about the cognitive and/or neural architecture assumed to underly cognitive capacities: If a cognitive capacity ψ is believed to be implemented by an architecture of type A, then the computational-level theory ψ_T is to be a function computable in fpt-time by at least one possible instantiation of architecture A. Such a refinement is not in contradiction to the FPT-Cognition thesis as I formulated it here, but merely helps shrink the space of feasible

cognitive theories, possibly leaving even fewer possibilities to consider than existed under the P-Cognition thesis without such architectural constraints. Given the inevitable empirical underdetermination of computational-level theories, this kind of refinement would make for a welcome adjustment.

It cannot be ruled out that future research exposes problems with the FPT-Cognition thesis or otherwise leads to the development of alternative, competing formalizations of the Tractable Cognition thesis. Here I considered the possibility of alternative formalizations based on parallel, probabilistic, or quantum computing models. Current mathematical research strongly suggests that the computational power of parallel and probabilistic computational machines is directly comparable to deterministic Turing machines. As for quantum computing models, there are strong arguments for why these models may be unrealistic as models of human cognition, and in any event the computational powers of quantum computers do not far exceed those of classical computing machines. In all, this means that it is unlikely that different choices in machine models alone will lead researchers to prefer alternative formalizations over the FPT-Cognition thesis. Still, other formalizations may prove better for other reasons, and only future research on the topic can tell. With this article, I hope to have laid the groundwork for future meta-theoretical and formal research investigating the Tractable Cognition thesis and its applications in cognitive science.

In the mean time, the tools and proof techniques described here can already be used by cognitive scientists to position their own theories relative to the two notions of tractability discussed in this article. Those cognitive scientists who agree with me that fpt-time computation is a reasonable upperbound on human cognitive computational powers can use the FPT-Cognition thesis to constrain and revise their computational-level theories as suggested in section 5.1. Doing so will not only allow these researchers to perform stronger empirical tests of their theories, but the resulting computational-level theories will have the theoretical strength that we can reasonably assume that algorithmic- and implementational-level explanations of the modelled capacities can one day exist.

Notes

1. The last is more of interest to artificial intelligence than to cognitive psychology.
2. Sometimes claims about super-Turing computing powers are made in the cognitive science literature without any reasonable argument or even a reference (e.g., van Gelder, 1999), and very often the distinction between computability (as discussed in this section and formalized by Turing) and computational tractability (as discussed in sections 3 and 5 in this article) is muddled or ignored (e.g., van Gelder, 1998). It is true that results about super-Turing computing can be found in the theoretical computer science literature (e.g., Siegelmann & Sontag, 1994). However, these results seem to depend crucially on the assumption of infinite precision (or infinite speed-up; e.g., Copeland, 2002), and thus the practicality of these results can be questioned. Furthermore, even if infinite precision is possible in some physical systems, it may still not be possible

in human cognitive systems (cf. Chalmers, 1994; Eliasmith, 2000, 2001). Last, even if infinite precision is possible for some variables in cognitive systems, the super-Turing computing powers do not necessarily extend when tractability constraints are imposed (see Siegelmann & Sontag, 1994).

3. Frixione (2001) used the term *Invariance thesis* to express both the Invariance thesis itself and what I call the Tractable Cognition thesis (Fig. 3). Because I intend to investigate two alternative formalizations of tractability in this article (viz., classical and fixed-parameter tractability), whereas Frixione (2001) only considered one option (classical tractability), I purposely divorce the Invariance thesis from the Tractable Cognition thesis.
4. The reader is advised that what I call classical complexity theory is typically referred to as (computational) complexity theory in both the computer science and cognitive science literature. Because I wish to contrast this theory with a newer form of complexity theory, called parameterized complexity theory, I refer to the earlier theory as "classical."
5. The classes P and NP are traditionally classes of decision problems: that is, input/output functions with only two possible outputs ("1" or "0," or "yes" or "no"). Because decision problems and search problems (such as the ones shown in Table 2) are closely related, we can ignore the distinction in this article. For more details on this relationship, refer to van Rooij (2003).
6. In this context, it is of interest to note that Thagard also proposed that his computational-level theory ψ_{Coh} explains why computer scientists believe that P $\neq$ NP (Thagard, 1993).
7. For example, it has been shown that $|C^-|$-ψ_{Coh} can be reduced to the polynomial-time computable problem Min-Cut (van Rooij, 2003), s-ψ_{Coh} can be reduced to the special case of s-ψ_{Coh} where the input size is bounded by the parameter, $|i| \leq g(s)$ (van Rooij, 2003); and u-ψ_{Coh} can be reduced to the known fixed-parameter tractable problem Edge-Bipartization (Stege & van Rooij, 2006).
8. It may be informative to note that Thagard's (2000) approach complies with the standard approach for dealing with intractability in algorithm design (which is a case of forward engineering), but as explained by van Rooij and Wareham (2008), this approach is not appropriate for cognitive modeling (which is a case of reverse engineering).
9. It may be interesting in this context to mention that settling in a maximum harmony activation pattern for a Hopfield network (see Table 2) is equivalent to computing the maximum coherence partition of propositions in ψ_{Coh}. This means that the fpt-results reported here generalize directly also to Network Harmony theories: Settling in a maximum harmony activation pattern in a Hopfield network is computationally tractable if overall network harmony is high or low, or when there are relatively few negatively weighted connections in the network.
10. When no resource bound is imposed, it is known that deterministic and non-deterministic computation is of equivalent power (see the Appendix, section A.3).
11. This parallel machine is purely fictive, and none such parallel machine can exist for all serial computations because it is known that some computations are not paralleliz-

able. The point is merely to illustrate some fundamental obstacles to tractable parallel computation of intractable serial computations.

12. In that sense, the expression "fixed-parameter tractability" may be a misnomer.

Acknowledgments

I gratefully acknowledge my Ph.D. advisors, Helena Kadlec and Ulrike Stege, whose guidance and support have been invaluable in the development of the ideas expressed in this article. I thank Stefan Frank, Moritz Müller, Paul Thagard, Todd Wareham, and three anonymous reviewers for helpful comments on earlier versions of this article; and Scott Aaronson for helpful feedback on my informal characterization of quantum computers and their computational limitations. Thanks also to the members of the Cognitive group and the PITA group at the University of Victoria, the JDM group at Eindhoven University of Technology, the EEC group and the Computational Modeling group at Radboud University Nijmegen, the ASCOT group at the University of Utrecht, and members of the Department of Computer Science at Memorial University of Newfoundland for useful discussions. Parts of this article also appeared in a doctoral dissertation submitted for a Ph.D. degree in psychology at the University of Victoria.

References

Anderson, J. R. (1978). Arguments concerning representations for mental imagery. *Psychological Review, 85*, 249–277.

Anderson, J. R. (1987). Methodologies for studying human knowledge. *Behavioral and Brain Sciences, 10*, 467–505.

Anderson, J. R. (1990). *The adaptive character of thought*. NJ: Lawrence Erlbaum Associates, Inc. Hillsdale.

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

Barton, G. E., Berwick, R. C., & Ristad, E. S. (1987). *Computational complexity and natural language*. Cambridge, MA: MIT Press.

Bylander, T. (1994). The computational complexity of propositional STRIPS planning. *Artificial Intelligence, 69*, 165–204.

Chalmers, D. J. (1994). *A computational foundation for the study of cognition (Tech. Rep. in Philosophy-Neuroscience-Psychology.)* Washington University. Available at http://consc.net/papers/computation.html

Chater, N., & Oaksford, M. (2000). The rational analysis of mind and behavior. *Synthese, 122*, 93–131.

Chater, N., Tenebaum, J. B., & Yuille, A. (Eds.). (2006). Special issue: Probabilistic models in cognition. [Special issue] *Trends in Cognitive Sciences, 10* (7).

Chater, N., & Vitányi, P. (2003a). Generalized law of universal generalization. *Journal of Mathematical Psychology, 47*, 346–369.

Chater, N., & Vitányi, P. (2003b). Simplicity: A unifying principle in cognitive science? *Trends in Cognitive Sciences, 7*, 19–22.

Cherniak, C. (1986). *Minimal rationality.* Cambridge, MA: MIT Press.

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

Cleland, C. E. (1993). Is the Church–Turing thesis true? *Minds and Machines, 3*, 283–312.

Cleland, C. E. (1995). Effective procedures and computable functions. *Minds and Machines, 5*, 9–23.

Cockshott, P., & Michaelson, G. (2005). Are there new models of computation? Reply to Wegner and Eberbach. *Computer Journal, 50*, 232–247.

Cook, S. (1971). The complexity of theorem-proving procedures. In M. A. Harrison, R. B. Banerji, and J. D. Ullman (Eds.), *Proceedings of the 3rd Annual ACM symposium on Theory of Computing* (pp. 151–158). New York: ACM Press.

Cooper, G. F. (1990). The computational complexity of probabilistic inference using Bayesian belief networks. *Artificial Intelligence, 42*, 393–405.

Copeland, B. J. (2002). Accelerating Turing machines. *Minds and Machines, 12*, 281–301.

Cummins, R. (2000). "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.

Deutsch, D. (1985). Quantum theory, the Church–Turing principle and the universal quantum computer. *Proceedings of the Royal Society of London A, 400*, 97–117.

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

Dry, M., Lee, M. D., Vickers, D., & Hughes, P. (2006). Human performance on visually presented traveling salesperson problems with varying numbers of nodes. *Journal of Problem Solving, 1*, 20–32.

Dunne, P. E., Laurence, M., & Wooldridge, M. (2003). Complexity results for agent design problems. *Annals of Mathematics, Computing & Teleinformatics, 1*(1), 19–36.

Eliasmith, C. (2000). Is the brain analog or digital? The solution and its consequences for cognitive science. *Cognitive Science Quarterly, 1*, 147–170.

Eliasmith, C. (2001). Attractive and in-discrete: A critique of two putative virtues of the dynamicist theory of mind. *Minds and Machines, 11*, 417–426.

Fellows, M. R. (2002). Parameterized complexity: The main ideas and connections to practical computing. In R. Fleischer, B. Moret, & E. Meineche Schmidt (Eds.), *Experimental algorithmics: From algorithm design to robust and efficient software* (pp. 51–77). Berlin: Springer-Verlag.

Fishburn, P. C., & LaValle, I. H. (1993). Subset preferences in linear and nonlinear utility theory. *Journal of Mathematical Psychology, 37*, 611–623.

Fishburn, P. C., & LaValle, I. H. (1996). Binary interactions and subset choice. *European Journal of Operational Research, 92*, 182–192.

Flum, J., & Grohe, M. (2006). *Parameterized complexity theory*. New York: Springer-Verlag.

Fodor, J. A. (1987). *Psychosemantics: The problem of meaning in the philosophy of mind*. Cambridge, MA: MIT Press.

Fortnow, L. (2003). One complexity theorist's view of quantum computing. *Theoretical Computer Science, 292*, 597–610.

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

Gandy, R. (1988). The confluence of ideas in 1936. In R. Herken (Ed.), *The universal Turing machine: A half-century survey* (pp. 55–111). New York: Oxford University Press.

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

Gibbons, A., & Rytter, W. (1988). *Efficient parallel algorithms*. Cambridge, England: Cambridge University Press.

Gigerenzer, G., & Goldstein, D. G. (1996). Reasoning the fast and frugal way: Models of bounded rationality. *Psychological Review, 103*, 650–669.

Grush, R., & Churchland, P. S. (1995). Gaps in Penrose's toilings. *Journal of Consciousness Studies, 2*, 10–29.

Hahn, U., Chater, N., & Richardson, L. B. (2003). Similarity as transformation. *Cognition, 87*, 1–32.

Hamilton, M., Müller, M., van Rooij, I., & Wareham, T. (2007). Approximating solution structure. In E. Demaine, G. Z. Gutin, D. Marx, and U. Stege (Eds.), *Proceedings of the Dagstuhl Seminar on Structure Theory and FPT Algorithmics for Graphs, Digraphs and Hypergraphs* (pp. 1–24). Internationales Begegnugs- und Forschungszentrum für Informatik (IBFI), Schloss Dagstuhl, Germany.

Harnad, S. (Ed.). (1987). *Categorical perception: The groundwork of cognition*. Cambridge, England: Cambridge University Press.

Haselager, W. F. G. (1997). *Cognitive science and folk psychology: The right frame of mind*. London: Sage.

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

Horsten, L., & Roelants, H. (1995). The Church–Turing thesis and effective mundane procedures. *Minds and Machines, 5*, 1–8.

Israel, D. (2002). Reflections on Gödel's and Gandy's reflections on Turing's thesis. *Minds and Machines, 12*, 181–201.

Jagota, A. (1997). Optimization by a Hopfield-style network. In D. S. Levine & W. R. Elsberry (Eds.), *Optimality in biological and artificial networks?* (pp. 203–226). Hillsdale, NJ: Lawrence Erlbaum Associates, Inc.

Jaja, J. (1992). *An introduction to parallel algorithms.* Reading, Addison, MA: Wesley.

Joseph, D. A., & Plantinga, W. H. (1985). On the complexity of reachability and motion planning problems. In J. O'Rourke (Ed.), *Proceedings of the 1st ACM symposium on Computational Geometry* (pp. 62–66). New York: ACM Press.

Judd, J. S. (1990). *Neural network design and the complexity of learning*. Cambridge, MA: MIT Press.

Kak, S. (2000). Active agents, intelligence, and quantum computing. *Information Sciences, 128*, 1–17.

Karp, R. M. (1972). Reducibility among combinatorial problems. In R. Miller & J. Thatcher (Eds.), *Complexity of computer computations* (pp. 85–104). New York: Plenum.

Kleene, S. C. (1936). General recursive functions of natural numbers. *Mathematische Annelen, 112*, 727–742.

Kleene, S. C. (1988). Turing's analysis of computability, and major applications of it. In R. Herken (Ed.), *The universal Turing machine: A half-century survey* (pp. 17–54). New York: Oxford University Press.

Krueger, L. E., & Tsav, C.-Y. (1990). Analyzing vision at the complexity level: Misplaced complexity? *Behavioral and Brain Sciences, 13*, 449–450.

Kube, P. R. (1991). Unbounded visual search is not both biologically plausible and NP-complete. *Behavioral and Brain Sciences, 14*, 768–773.

Levesque, H. J. (1988). Logic and the complexity of reasoning. *Journal of Philosophical Logic, 17*, 355–389.

Lewis, H. R., & Papadimitrou, C. H. (1998). *Elements of the theory of computation.* Upper Saddle River, NJ: Prentice Hall.

Li, M., & Vitányi, P. (1997). *An introduction to Kolmogorov complexity and its applications.* New York: Springer-Verlag.

Litt, A., Eliasmith, C., Kroon, F. W., Weinstein, S., & Thagard, P. (2006). Is the brain a quantum computer? *Cognitive Science, 30*, 593–603.

Marr, D. (1982). *Vision: A computational investigation into the human representation and processing visual information*. San Francisco: Freeman.

Martignon, L., & Hoffrage, U. (2002). Fast, frugal, and fit: Simple heuristics for paired comparison. *Theory and Decision, 52*, 29–71.

Martignon, L., & Schmitt, M. (1999). Simplicity and robustness of fast and frugal heuristics. *Minds and Machines, 9*, 565–593.

Massaro, D. W., & Cowan, N. (1993). Information processing models: Microscopes of the mind. *Annual Review of Psychology, 44*, 383–425.

Millgram, E. (2000). Coherence: The price of the ticket. *Journal of Philosophy, 97*, 82–93.

Narayanan, A. (1999). Quantum computing for beginners. *Proceedings of the 1999 Congress on Evolutionary Computation* (pp. 2231–2238). Piscataway, NJ: IEEE Press.

Nebel, B. (1996). Artificial intelligence: A computational perspective. In G. Brewka (Ed.), *Principles of knowledge representation* (pp. 237–266). Stanford, CA: CSLI Publications.

Newell, A., & Simon, H. A. (1988a). GPS, a program that simulates human thought. In A. M. Collins & E. E. Smith (Eds.), *Readings in cognitive science: A perspective from psychology and artificial intelligence* (pp. 453–460). San Mateo, CA: Kaufmann.

Newell, A., & Simon, H. A. (1988b). The theory of human problem solving. In A. M. Collins & E. E. Smith (Eds.), *Readings in cognitive science: A perspective from psychology and artificial intelligence* (pp. 33–51). San Mateo, CA: Kaufmann.

Niedermeier, R. (2006). *Invitation to fixed-parameter algorithms*. New York: Oxford University Press.

Oaksford, M., & Chater, N. (1993). Reasoning theories and bounded rationality. In K. I. Manktelow & D. E. Over (Eds.), *Rationality: Psychological and philosophical perspectives* (pp. 31–60). London: Routledge.

Oaksford, M., & Chater, N. (1998). *Rationality in an uncertain world: Essays on the cognitive science of human reasoning*. Hove, England: Psychology Press.

Papadimitriou, C. H., & Steiglitz, K. (1988). *Combinatorial optimization: Algorithms and complexity.* New York: Dover.

Parberry, I. (1994). *Circuit complexity and neural networks*. Cambridge, MA: MIT Press.

Parberry, I. (1997). Knowledge, understanding, and computational complexity. In D. S. Levine & W. R. Elsberry (Eds.), *Optimality in biological and artificial networks?* (pp. 125–144). Mahwah, NJ: Lawrence Erlbaum Associates, Inc.

Penrose, R. (1989). *The emperor's new mind: Concerning computers, minds, and the laws of physics*. New York: Oxford University Press.

Penrose, R. (1994). *Shadows of the mind*. Oxford: Oxford University Press.

Penrose, R. (1997). Physics and the mind. In M. Longair (Ed.), *The large, the small and the human mind* (pp. 93–143). Cambridge, England: Cambridge University Press.

Pizlo, Z., Stefanov, E., Saalweachter, J., Li, Z., Haxhimusa, Y., & Kropatsch, W. G. (2006). Traveling salesman problem: A foveating pyramid model. *Journal of Problem Solving, 1*, 83–101.

Port, R. F., & van Gelder, T. (Eds.), (1995). *Mind as motion: Explorations in the dynamics of cognition.* Cambridge, MA: MIT Press.

Post, E. L. (1936). Finite combinatory processes-formulation I. *Journal Symbolic Logic, 1*, 103–105.

Pothos, E. M., & Chater, N. (2001). Basic categories by simplicity. In U. Hahn & M. Ramscar (Eds.), *Similarity and categorization* (pp. 51–72). Oxford England: Oxford University Press.

Pothos, E. M., & Chater, N. (2002). A simplicity principle in unsupervised human categorization. *Cognitive Science, 26*, 303–343.

Prasse, M., & Rittgen, P. (1998). Why Church's thesis holds: Some notes on Peter Wegner's tracts on interaction and computability. *Computer Journal, 41*, 357–362.

Pylyshyn, Z. (1984). *Computation and cognition: Towards a foundation for cognitive science.* Cambridge, MA: MIT Press.

Pylyshyn, Z. (1991). The role of cognitive architectures in the theory of cognition. In K. VanLehn (Ed.), *Architectures for intelligence* (pp. 189–223). Hillsdale, NJ: Lawrence Erlbaum Associates, Inc.

Reiter R. (1980). A logic for default reasoning. *Artificial Intelligence, 13*, 81–132.

Rensink, R. A., & Provan, G. (1991). The analysis of resource-limited vision systems. In K. J. Hamrnond and D. Gentner (Eds.), *Proceedings of the 13th annual conference of the Cognitive Science Society* (pp. 311–316). Hillsdale, NJ: Erlbaum.

Ristad, E. S. (1990). *Computational structure of human language*. Unpublished doctoral thesis, Department of Electrical Engineering and Computer Science, MIT, Cambridge, MA.

Ristad, E. S. (1993). *The language complexity game*. Cambridge, MA: MIT Press.

Rosch, E. (1973). On the internal structure of perceptual and semantic categories. In T. E. Moore (Ed.), *Cognitive development and the acquisition of language* (pp. 111–144). New York: Academic.

Rosch, E., & Mervis, C. B. (1975). Family resemblances: Studies in the internal structure of categories. *Cognitive Psychology, 7*, 573–605.

Roth, D. (1996). On the hardness of approximate reasoning. *Artificial Intelligence, 82*, 273–302.

Rumelhart, D. E., McClelland, J. L., & the PDP Research Group. (1986). *Parallel distributed processing. Explorations in the microstructure of cognition. Volume 1: Foundations*. Cambridge, MA: MIT Press.

Shor, P. (1997). Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer. *SIAM Journal on Computing, 26*, 1484–1509.

Siegel, R. M. (1990). Is it really that complex? After all, there are no green elephants. *Behavioral and Brain Sciences, 13*, 453.

Siegelmann, H., & Sontag, E. (1994). Analog computation via neural networks. *Theoretical Computer Science, 131*, 331–360.

Simon, H. A. (1957). *Models of man: Social and rational*. New York: Wiley.

Simon, H. A. (1988). Rationality as process and as product of thought. In D. E. Bell, H. Raiffa, & A. Tversky (Eds.), *Decision making: Descriptive, normative, and prescriptive interactions* (pp. 58–77). Cambridge, England: Cambridge University Press.

Simon, H. A. (1990). Invariants of human behavior. *Annual Review of Psychology, 41*, 1–19.

Smolensky, P., & Legendre, G. (2006). *The harmonic mind: From neural computation to optimality-theoretic grammar.* Cambridge, MA: Bradford.

Stege, U., & van Rooij, I. (2006). *Computing maximum coherence: A hard nut to crack?* Paper presented at the *39th annual meeting of the Society for Mathematical Psychology*, Vancouver, British Columbia, July.

Steinhart, E. (2002). Logically possible machines. *Minds and Machines, 12*, 259–280.

Thagard, P. (1993). Computational tractability and conceptual coherence: Why do computer scientists believe that P $\neq$ NP? *Canadian Journal of Philosophy, 23*, 349–364.

Thagard, P. (2000). *Coherence in thought and action.* Cambridge, MA: MIT Press.

Thagard, P., & Verbeurgt, K. (1998). Coherence as constraint satisfaction. *Cognitive Science, 22*, 1–24.

Thelen, E., & Smith, L. B. (1994). *A dynamic systems approach to the development of cognition and action.* Cambridge, MA: MIT Press.

Todd, P. M., & Gigerenzer, G. (2000). Précis of simple heuristics that make us smart. *Behavioral and Brain Sciences, 23*, 727–780.

Tsotsos, J. K. (1990). Analyzing vision at the complexity level. *Behavioral and Brain Sciences, 13*, 423–469.

Tsotsos, J. K. (2001). Complexity, vision and attention. In L. Harris and M. Jenkin (Eds.), *Vision & attention* (pp. 105–128). New York: Springer-Verlag.

Turing, A. M. (1936). On computable numbers, with an application to the Entscheidungsproblem. *Proceedings of the London Mathematical Society, 42*, 230–265.

Tversky, A. (1977). Features of similarity. *Psychological Review, 84*, 327–352.

van der Helm, P. A. (2004). Transparallel processing by hyperstrings. *Proceedings of the National Academy of Sciences, USA, 101*(30), 10862–10867.

van der Helm, P. A., & Leeuwenberg, E. L. J. (1986). Avoiding explosive search in automatic selection of simplest pattern codes. *Pattern Recognition, 19*, 181–191.

van der Helm, P. A., & Leeuwenberg, E. L. J. (1996). Goodness of visual regularities: A nontransformational approach. *Psychological Review, 103*, 429–456.

van Emde Boas, P. (1990). Machine models and simulations. In J. van Leeuwen, (Ed.), *Handbook of theoretical computer science, Volume A: Algorithms and complexity* (pp. 1–66). MIT Press, Cambridge: MA.

van Gelder, T. (1995). What might cognition be if not computation? *Journal of Philosophy, 7*, 345–381.

van Gelder, T. (1998). The dynamical hypothesis in cognitive science. *Behavioral and Brain Sciences, 21*, 615–665.

van Gelder, T. (1999). Defending the dynamical hypothesis. In W. Tschacher & J.-P. Dauwalder (Eds.), *Dynamics, synergetics, autonomous agents: Nonlinear systems approaches to cognitive psychology and cognitive science* (pp. 13–28). Singapore: World Scientific.

van Rooij, I. (2003). *Tractable cognition: Complexity theory in cognitive psychology.* Unpublished doctoral thesis, Department of Psychology, University of Victoria, British Columbia, Canada.

van Rooij, I., Stege, S., & Kadlec, H. (2005). Sources of complexity in subset choice. *Journal of Mathematical Psychology, 49*, 160–187.

van Rooij, I., & Wareham, T. (2008). Parameterized complexity in cognitive modeling: Foundations, applications, and opportunities. *Computer Journal, 51*, 385–404.

van Rooij, I., & Wright, C. (2006). The incoherence of heuristically explaining coherence. In R. Sun and N. Miyake (Eds.), *Proceedings of the 28th annual conference of the Cognitive Science Society* (p. 2622). Mahwah, NJ: Lawrence Erlbaum Associates, Inc.

Wareham, T. (1996). The role of parameterized computational complexity theory in cognitive modeling. In C. X. Ling and R. Sun (Eds.), *Working Notes of the AAAI-96 Workshop on Computational Cognitive Modeling: Source of the Power*.

Wareham, T. (1999). *Systematic parameterized complexity analysis in computational phonology.* Unpublished doctoral thesis, Department of Computer Science, University of Victoria, British Columbia, Canada.

Wareham, T. (2001). The parameterized complexity of intersection and composition operations on sets of finite-state automata. In S. Yu and A. Paun (Eds.), *Proceedings of the 5th International Conference on Implementation and Application of Automata* (pp. 302–310). Berlin: Springer-Verlag.

Wegner, P. (1997). Why interaction is more powerful than algorithms. *Communications of the ACM, 40*, 80–91.

Wegner, P., & Eberbach, E. (2004). New models of computation. *Computer Journal, 47*, 4–9.

Wells, A. J. (1998). Turing's analysis of computation and theories of cognitive architecture. *Cognitive Science, 22*, 269–294.

Wooldridge, M., & Dunne, P. E. (2005). The complexity of agent design problems: Determinism and history dependence. *Annals of Mathematics and Artificial Intelligence, 45*, 343–371.

Appendix

A.1. The intuitive notion of a computation

Informally, when we say a system computes an input/output function, $\psi : I \rightarrow O$, we mean to say that the system reliably transforms every $i \in I$ into $\psi(i) \in O$ in a way that can be described by an algorithm (In the literature, an algorithm is also sometimes referred to as mechanical procedure, effective procedure, or computation procedure). An algorithm is a step-by-step finite procedure that can be performed, by a human or machine, without the need for any insight, just by following the steps as specified by the algorithm. The notion of an algorithm, so described, is an intuitive notion. Mathematicians and computer scientists have pursued several formalizations (e.g., Church, 1936; Kleene, 1936; Post, 1936; Turing, 1936), all of which are known to be extensionally equivalent. Probably the best-known formalization, in particular among cognitive scientists and psychologists, is the one by Alan Turing. One of the strengths of Turing's formalization is its intuitive appeal and its simplicity.

Turing motivated his formalization by considering a paradigmatic example of computation: the situation in which a human sets out to compute a number using pen and paper (see Turing, 1936, pp. 249–252). Turing argued that a human computer can be in, at most, a finite number of different "states of mind" because if "we admitted an infinity of states of mind, some of them will be 'arbitrarily close' and will be confused" (p. 250). Similarly, Turing argued a human computer can read and write only a finite number of different symbols because if "we were to allow an infinity of symbols, then there would be symbols different to an arbitrarily small extent" (p. 249). On the other hand, Turing allowed for a potentially infinite paper resource. He assumed that the paper is divided into squares (like an arithmetic notebook) and that symbols are written in these squares. With respect to the reading of symbols, Turing wrote: "We may suppose that there is a bound B on the number of symbols or squares that the computer can observe at one moment. If [she or he] wishes to observe more [she or he] must use successive operations" (p. 250). This restriction was motivated by the observation that for long lists of symbols, we cannot tell them apart in "one look." Compare, for example, the numbers 96785959943 and 96785959943. Are they the same or different?

According to Turing (1936) the behavior of a human computer at any moment in time is completely determined by his or her state of mind and the symbol(s) she or he is observing. The computer's behavior can be understood as a sequence of operations, with each operation "so elementary that it is not easy to imagine [it] further divided" (p. 250). Turing distinguished the following two elementary operations:

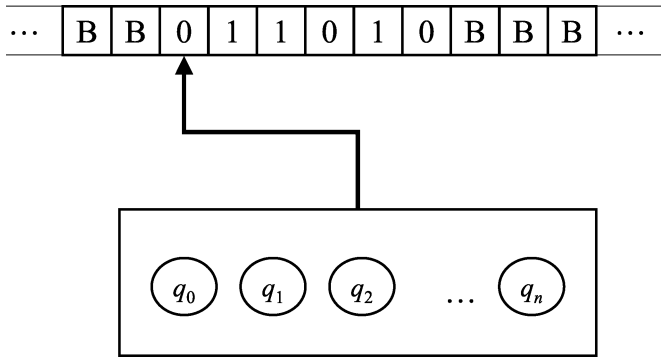

Fig. A1. Illustration of a Turing machine. The tape extends left and right into infinity. Each square on the tape contains a symbol in the set $\{1, 0, B\}$. The machine can read and write symbols with a read–write head, and can be in a finite number of different machine states $\{q_0, q_1, q_2, \ldots, q_n\}$.

(a) A possible change of a symbol on an observed square.

(b) A possible change in an observed square.

Each operation is followed by a possible change in state of mind. With this characterization of computation, Turing could define a machine to do the work of a human computer. Fig. A1 illustrates this machine.

A.2. The Turing machine formalism

A Turing machine M is a machine that at any moment in time is in one of a finite number of machine states (analogue to "states of mind"). The set of possible machine states is denoted by $Q = \{q_0, q_1, q_2, \ldots, q_n\}$. One machine state, q_0, is designated the initial state; this is the state that M is in at the beginning of the computation. There is also a non-empty set, $H \subset Q$, of halting states; whenever the machine goes into a state, $q_i \in H$, then the machine halts and the computation is terminated.

The machine has a read–write head that gives it access to an external memory, represented by a one-dimensional tape (analogue to the paper). The tape is divided in discrete regions called tape squares. Each tape square may contain, at most, one symbol. The machine can move the read–write head from one square to a different square, always moving the read–write head to the right or left, at most, one square at a time. The read–write head is always positioned on one (and, at most, 1) square, which it is said to scan. If a square is scanned, then the machine can read a symbol from or write a symbol to that square. At most one symbol can be read or written at a time.

The set of possible symbols is denoted by S, and is called the alphabet of M. S is a finite set. Often it is assumed that $S = \{0, 1, B\}$, where B is called the blank. Time is discrete for M, and time instants are ordered $0, 1, 2, \ldots$. At time 0, the machine is in its initial state, q_o, the read–write head is in a starting square, and all squares contain Bs except for a finite sequence of adjacent squares, each containing either 1 or 0. The sequence of 1s and 0s on the tape at time 0 is a called the input.

The Turing machine can perform two types of basic operations:

(a') It can write an element from S in the square it scans.

(b') It can shift the head one square left (L) or right (R).

After performing an operation of either type, (a') or (b'), the machine takes on a state in Q. At any one time, which operation is performed and which state is entered is completely determined by the present state of the machine and the symbol presently scanned. In other words, the behavior of a Turing machine can be understood as being governed by a function T that maps a subset of $Q \times S$ into $A \times Q$, where $A = \{0, 1, B, L, R\}$ denotes the set of possible operations.

We call T the transition function of M. A transition $T(p, s) = (a, q)$ is interpreted as follows: If $p \in Q$ is the current state and $s \in S$ is the current scanned symbol, then the machine performs operation $a \in A$ of type (a') or (b'), and the machine enters the state $q \in Q$. For example, $T(p, 0) = (1, q)$ means that if M is in state p and read symbol 0, then M is to write symbol 1 and go into state q; $T(p, 1) = (L, q)$ means that if M is in state p and reads symbol 1, then M is to move its read–write head one square to the left and go into state q. Note that Q, S, and A are finite sets. Thus we can also represent the transition function T as a finite list of transitions. Such a list is often called the machine table, and transitions are then called machine instructions.

Under the governance of T, the machine M performs a uniquely determined sequence of operations, which may terminate in a finite number of steps. If the machine does halt then the sequence of symbols on the tape is called its output. A Turing machine is said to compute a function $\psi : I \to O$ if for every possible input $i \in I$ it outputs $\psi(i)$. A function is called *computable* (or *Turing-computable*) if there exists a Turing machine that computes it. Turing (1936) proved that there exist (infinitely many) problems that are not computable. For example, he showed that the Halting problem is not computable. This decision problem is formulated as follows:

Halting problem
Input: A Turing machine M and an input i for M.
Question: Does M halt on i?

A.3. Extensions of the Turing Machine Concept

The reader may wonder to what extent the particular limitations placed by Turing on his machine are crucial for the limitations on computability. Therefore, a few notes should be made on the computational power of the Turing machine with certain extensions. It has been shown that several seemingly powerful adjustments to Turing's machine do not increase its computational power (see, e.g., Lewis & Papadimitriou, 1998, for an overview). For example, the set of functions computable by the Turing machine described above is the same as the set of functions computable by Turing machines with one or more of the following extensions:

1. Turing machines with multiple tapes and multiple read–write heads.
2. Turing machines with any finite alphabets (i.e., not necessarily $A = \{0, 1, B\}$).

3. Turing machines with random access: These are Turing machines that can access any square on the tape in a single step.
4. Non-deterministic Turing machines: These are Turing machines that, instead of being governed by a transition function, are governed by a transition relation, mapping some elements in $Q \times S$ to possibly more than one element in $A \times Q$. Such a non-deterministic machine is said to "compute" a function ψ if for every input i there exist one possible sequence of operations that, when performed, would lead to output $\psi(i)$.

It should be noted that machines of Type 4 are not considered to really compute in the sense that Turing meant to capture with his formalism; namely, in non-deterministic machines, not every step of the computation is uniquely determined, and thus, a human wishing to follow the set of instructions defined by the machine table would not be able to unambiguously determine how to proceed at each step. Even though non-deterministic machines are purely theoretical constructs they do serve a special purpose in theories of computational intractability (see section 3). The extensions (1)–(3), on the other hand, are considered reasonable extensions. Throughout this article, the term *Turing machine* can be taken to refer to Turing machines both with and without such extensions.