Logic and Complexity in Cognitive Science

Alistair Isaac[1] and Jakub Szymanik[2]

[1]Department of Philosophy, University of Michigan
[2]Institute of Artificial Intelligence, University of Groningen

March 7, 2011

1 Introduction

How can logic help us to understand cognition? One answer is provided by the computational perspective, which treats cognition as information flow in a computational system. This perspective draws an analogy between intelligent behavior as we observe it in human beings and the complex behavior of man-made computational devices, such as the digital computer. If we accept this analogy, then the behavior of cognitive systems in general can be investigated formally through logical analysis. From this perspective, logical methods can analyze:

1. the boundary between possible and impossible tasks
2. the efficiency with which any possible task can be solved
3. the low-level information flow which implements a solution

This paper will survey some examples of the application of logical techniques in each of these areas.

In general, we will see a back and forth between logical analysis and empirical findings. This back and forth helps to bridge the gap between the normative and descriptive roles of logic. For example, we antecedently believe humans should perform a certain way on a given task because that is the logical thing to do. We observe that they do not, in fact, perform as predicted. This does not change our assessment that human behavior can be described in terms of logical operations, but it changes our analysis of which task exactly humans perform in response to a particular experimental setup. The Wason Selection task provides an example of this sort (Section 3.1).

Likewise, suppose we analyze the complexity of a task and determine that it has no efficient solution. If we observe humans apparently solving this task, we use this analysis as evidence that they are in fact solving a

different, simpler problem. For example, the complexity of exhaustive visual search motivates the idea that the search space is constrained by top-down information (Tsotsos, 1990). More generally, complexity analysis can make predictions about the relationship between input size and solution speed for particular tasks. These predictions can be compared with empirical evidence to determine when subjects switch from one algorithm to another, as in the counting of objects in the visual field or quantifier processing (Section 6.2).

Given the ubiquity of logic and its flexibility as a tool for analyzing complex systems, we do not presume to cover all possible roles of logic in cognitive science.[1] However, focusing on the computational perspective highlights two properties of logical analysis essential for cognitive science. On the one hand, it clarifies conceptual debates by allowing them to be addressed in a precise manner (see, for example, its role in the symbolic / connectionist debate in Section 4.2). On the other hand, it can drive empirical research by providing specific predictions. The convergence of these two roles is best exemplified by complexity analysis, which can clarify the properties of proposed information processing tasks in an abstract but precise manner. Vague a priori debates about the nature of cognition can be tied to specific empirical predictions, thereby focusing them upon their real world consequences. Finally, complexity analysis and representation theorems help to bridge the gap between top-down task analysis and the bottom-up investigation of neural wiring, a critical research area for the current stage of cognitive science.

Section 2 outlines the basic features of the computational perspective. Section 3 looks at the apparent breakdown between human behavior and logical reasoning. It introduces non-monotonic logic as a means of more realistically analyzing cognitive behavior in a complex world. In Section 4, the connection between logical structure and neural wiring is addressed. In particular, we discuss how representation theorems undermine the supposed tension between symbolic and connectionist approaches to cognitive science. Though they may focus on different levels of cognitive organization, there is no *in principle* difference in the types of problems the two approaches can analyze.

Section 5 introduces the basics of computational complexity theory. This provides a second perspective from which the supposed conflict between different modeling formalisms can be dissolved. Finally, Section 6 demonstrates the application of complexity analysis to a variety of issues in cognitive science, concluding with a discussion of objections to this approach.

[1] For more references on the interface between logic and cognition, see also the 2007 special issue of *Topoi* on "Logic and Cognition", edited by Johan van Benthem and Helen and Wilfrid Hodges, the 2008 special issue of *Journal of Logic, Language and Information* "Formal Models for Real People" edited by Marian Counihan, and the 2008 special issue of *Studia Logica* on "Psychologism in Logic?", edited by Hannes Leitgeb.

2 The Computational Perspective

Alan Turing (1950) proposed that the true test of machine intelligence is indistinguishability from human intelligence and suggested a concrete method for determining if this criterion is satisfied. A human judge sits in front of two terminals, one allows him to communicate with a computer and the other one with a human being. The judge can type whatever text he chooses into the terminals. His task is to use the answers he receives to decide which terminal is connected to a human and which to a machine. If the judge cannot tell the difference, i.e. if he performs at chance in distinguishing human from computer, the computer has passed the Turing test.[2]

Turing's test assumes the Church-Turing thesis, which states that all computation (in the intuitive sense of a mechanical procedure for solving problems) is formally equivalent to Turing computation (computation by a Turing machine). This is a conceptual claim which cannot be formally proved. However, all attempts so far to explicate intuitive computability (many of them independently motivated) have turned out to define exactly the same class of problems. For instance, definitions via abstract machines (random access machines, quantum computers, cellular automata, genetic algorithms), formal systems (the lambda calculus, Post rewriting systems), and particular classes of function (recursive functions) are all formally equivalent to the definition of Turing machine computability. These results provide compelling support for the claim that all computation is equivalent to Turing computation (see e.g. Cooper, 2003, or his paper in this volume for more details).

Moreover, the Church-Turing Thesis provides a precise analysis of the intuitive concept of an *algorithm*: a mechanical procedure that performs a calculation. Consequently, it allows us to conclude that a problem is not mechanically decidable from a proof that it is not Turing computable. Take the class of all functions from natural numbers to natural numbers. There are uncountably many such functions, but there are only countably many Turing machines. Hence, some functions on the natural numbers must not be computable. The most famous non-computable problems are the Halting Problem (decide whether Turing machine M will halt on input x), the decision problem for first-order logic (i.e., the question whether a given formula φ is a theorem), and the Tenth Hilbert Problem (the algorithmic solvability in integers of Diophantine equations).

[2] The Loebner Prize for artificial intelligence is an annual competition which awards cash prizes to the computers which come closest to passing the Turing test. Due to the difficulty of this task, various simplifications are implemented in order to make the test more feasible. At one point, for example, computer participants were allowed to specialize in a particular subject matter. Currently, any topic is fair game for discussion, but the judge is allowed a limited time to question participants. In 2011, the judge will be allowed 25 minutes interaction time total with the two participants, and 5 minutes for assessing which he or she thinks is human. See: http://www.loebner.net/Prizef/loebner-prize.html

If we accept that the human mind is a physical system, and we accept the Church-Turing Thesis, then we should also accept its psychological counterpart:

The human mind can only solve computable problems.

In other words, cognitive tasks comprise computable functions. From an abstract perspective, a cognitive task is an information-processing task. Given some input (e.g. a visual stimulus, a state of the world, a sensation of pain), produce an appropriate output (e.g. perform an action, draw a conclusion, utter a response). Generally, then, cognitive tasks can be understood as functions from inputs to outputs, and the psychological version of the Church-Turing Thesis states that the only realistic candidates for information processing tasks performed by the human mind are computable functions.

Not everyone accepts the psychological version of the Church-Turing Thesis. In particular, some critics have argued that cognitive systems can do more than Turing machines. For example, learning understood as identifiability in the limit (Gold, 1967) is not computable (see Kugel, 1986, for an extensive discussion). Another strand of argumentation is motivated by Gödel's theorems. The claim is that Gödel's incompleteness results somehow demonstrate that the human mind cannot have an algorithmic nature. For example, J.R. Lucas (1961) claimed:

> Goedel's theorem seems to me to prove that Mechanism is false, that is, that minds cannot be explained as machines. So also has it seemed to many other people: almost every mathematical logician I have put the matter to has confessed to similar thoughts, but has felt reluctant to commit himself definitely until he could see the whole argument set out, with all objections fully stated and properly met. This I attempt to do.

Then he gives the following argument: A computer behaves according to a program, hence we can view it as a formal system. Applying Gödel's theorem to this system we get a true sentence which is unprovable in the system. Thus, the machine does not know that the sentence is true while we can see that it is true. Hence, we cannot be a machine. Lucas' argument was revived by Roger Penrose (1994) who supplemented it with the claim that quantum properties of the brain allow it to solve uncomputable problems. Lucas' argument has been strongly criticized by logicians and philosophers (e.g. Benacerraf, 1967; Pudlak, 1999), as has Penrose's (e.g. Feferman, 1995).

If identifiability in the limit is the correct analysis of learning, then we must accept the possibility of hyper-computation, i.e. the physical realization of machines strictly more powerful than the Turing machine. Likewise with the arguments from Gödel's theorems: they each imply that the brain

instantiates a device capable of "super-Turing" computation. Examples of such strong machines have been explored theoretically, for instance Zeno-machines (Accelerated Turing machines), which allow a countably infinite number of algorithmic steps to be performed in finite time (see e.g. Syropoulos, 2008), and Analog Neural Networks, which allow computation over arbitrarily precise real values (e.g. Siegelmann, 1995, 2003). However, no plausible account of how such devices could be physically realized has ever been offered. Both Penrose's appeal to quantum properties of the brain and Siegelmann's arbitrarily precise neural networks fail to take into account the noise inherent in any real world analog system.[3] Once we take into account the electrical noise inherent in neural behavior (and, indeed, the noise inherent in any chaotic physical system), the precision required to produce stronger than Turing computation seems unlikely to be physically realized.

However, there are two more interesting reasons to endorse the psychological version of the Church-Turing Thesis than simple physical plausibility. The first is its fruitfulness as a theoretical assumption. If we assume that neural computability is equivalent to Turing computability, we can generate precise hypotheses about which tasks the human mind can and cannot perform. The second is the close concordance between the computational perspective and psychological practice. Experimental psychology is naturally task oriented, because subjects are typically studied in the context of specific experimental tasks. Furthermore, the dominant approach in cognitive psychology is to view human cognition as a form of information processing (see e.g. Sternberg, 2002). Once the general information processing perspective has been taken, however, a natural extension is an attempt to reproduce human behavior using computational models. Although much of this work employs Bayesian or stochastic methods[4] (rather than logic-based formalisms), it is predicated on the assumption of the psychological version of the Church-Turing Thesis. Furthermore, the continued success of this research program vindicates the assumption that the human brain is a Turing-strength computational device.

If we assume the psychological version of the Church-Turing Thesis, what

[3]Siegelmann repeatedly appeals to a result in Siegelmann and Sontag (1994) when arguing in later papers that analog neural networks do not require arbitrary precision (and are thus physically realizable). In particular, Lemma 4.1 shows that for every neural network which computes over real numbers, there exists a neural network which computes over truncated reals (i.e. reals precise only to a finite number of digits). However, the length of truncation required is a function of the length of the computation—longer computations require longer truncated strings. Consequently, if length of computation is allowed to grow arbitrarily, so must the length of the strings of digits over which the computation is performed in a truncated network. Thus, one still must allow computation over arbitrarily precise reals if one is considering the computational properties of analog neural networks *in general*, i.e. over arbitrarily long computation times.

[4]For a recent overview, see e.g. the 2006 special issue of *Trends in Cognitive Sciences* (vol. 10, no. 7) on probabilistic models of cognition.

does it tell us about how to analyze cognition? David Marr (1983) proposed a general framework for explanation in cognitive science based on the computational perspective. He argued that any particular task computed by a cognitive system must ultimately be analyzed at three levels (in order of decreasing abstraction):

1. the computational level (the problem solved or function computed);
2. the algorithmic level (the algorithm used to achieve a solution);
3. the implementation level (how the algorithm is actually implemented in neural activity).

Considerations at each of these levels may constrain answers at the others, although Marr argued that analysis at the computational level is the most critical for achieving progress in cognitive science (Marr, 1983, p. 27).

Marr's three level system can only be applied *relative to* a particular computational question. For instance, a particular pattern of neural wiring may implement an algorithm which performs the computational function of detecting edges at a particular orientation. But of each neuron in that pattern, we can ask what is its computational function (usually, to integrate over inputs from other neurons) and how is this function implemented (electrochemical changes in the cell). Likewise, we may take a detected edge as an informational primitive when analyzing a more high-level visual function, such as object identification. Nevertheless, the most obvious examples of computational level analysis concern human performance on behavioral tasks, and the obvious target for implementation level analysis is neural wiring. Algorithmic analysis via complexity theory can then play the crucial role of bridging the gap between these two domains.

In the remainder of this paper, we examine how logic can play a role in the investigation of Marr's three levels. Although logical analysis can be constructive at each of these levels, the real power of logic can be seen in how it bridges the grey areas between levels. Representation theorems, for example, can bridge the gap between the algorithmic and implementation levels by providing an abstract description of the processing steps a physical system can perform. A language which can describe these steps is more abstract than any particular physical system (and hence is multiply realizable), yet it does not fully specify an algorithm, it merely constrains the types of algorithms allowable (Section 4). Likewise, complexity analysis can bridge the gap between the computational and algorithmic levels by specifying properties of the class of all algorithms which solve a particular task, e.g. the efficiency with which they operate. Such analysis can produce precise predictions about which tasks are involved in producing a particular behavior, and how reaction times will grow with length of input (Section 6). Before examining the interstices in Marr's levels, however, let's look at the

most abstract level on which human behavior can be analyzed and ask "Do human's behave logically?"

3 The Computational Level: Human Behavior

A number of results from experimental psychology seem to indicate that humans do not behave in accordance with the recommendations of ideal rationality. For example, they do not always maximize utility (e.g. in the ultimatum game), they do not reason in accordance with the rules of probability (e.g. the conjunction fallacy), and they do not follow the rules of classical logic (e.g. the Wason selection task). Since logic appears to provide the most fundamental norms for human reasoning (norms recognized by Aristotle thousands of years before those of decision theory or probability), human violation of logical norms provides the most distressing challenge to the assumption of human rationality.

It is worth noting, however, that many other violations of ideal rationality ultimately reduce to logical violations. Consider, for example, the conjunction fallacy: after reading a short passage about Linda which describes her as a social activist in college, 85% of subjects reported that the proposition that "Linda is a bank teller and is active in the feminist movement" was more probable than the proposition that "Linda is a bank teller" (Tversky and Kahneman, 1983). This is a fallacy because the axioms of probability ensure that $P(A\&B) \leq P(A)$ for all A and B. Yet this consequence of probability theory is itself a consequence of the basic set theoretical fact that $A \cap B \subseteq A$, which, under the standard semantics of propositional logic, is equivalent to $A\&B \rightarrow A$, a basic axiom.

From the perspective of cognitive science, the apparent irrationality of human behavior is only problematic insofar as it defeats our attempts to provide a computational analysis of the task being performed. In the case of the Wason selection task, apparently irrational behavior drove the development of increasingly sophisticated computational models. After discussing this specific example, we'll look at the Frame problem, a more general challenge to the computational perspective. This problem motivated the development of non-monotonic logic as a means of providing a formal analysis of human reasoning. This tool will also prove useful when we examine the properties of neural wiring in the next section.

3.1 Human Behavior is [not?] Logical

The Wason selection task (Wason, 1968; Wason and Shapiro, 1971) seems to indicate that humans are very poor at applying even simple rules of reasoning such as modus tollens. Furthermore, Cheng et al. (1986) suggests they may continue to be poor even when they have taken an introductory logic class! Does this imply that human behavior does not decompose into

logical steps? Or that our neural wiring is somehow qualitatively different from the logical structure which can be found in any computational device simulating the brain?

The original Wason selection task is very simple. Subjects are shown four cards and told that all cards have numbers on one side and letters on the other. The faces visible to the subject read D, K, 3, and 7. The subject is then told "Every card which has a D on one side has a three on the other" and asked which cards they need to turn over to verify this rule. From a classical standpoint, the claim has the basic structure of a material conditional, $D \rightarrow 3$, and the correct answer is to turn over cards D and 7. However, the most popular answers (in order of decreasing popularity) are (1) D and 3; (2) D; (3) D, 3, and 7; (4) D and 7. The classically correct answer ranks fourth, while an instance of affirming the consequent (i.e. judging that 3 is relevant for determining if the rule is correct) ranks first. Wason's robust and frequently reproduced result seems to show that most people are poor at modus tollens and engage in fallacious reasoning on even very simple tasks. Are we really this bad at logic?

As it turns out, there are complexities in the data. The original selection task involved an abstract domain of numbers and letters. When the problem is rephrased in terms of certain types of domain with which subjects are familiar, reasoning suddenly improves. For example, Griggs and Cox (1982) demonstrate that if cards have ages on one side and types of drink on the other, subjects perform near perfectly (i.e. in accordance with the classical recommendation) when the task is to determine which cards to turn over to ensure that the rule "if a person is drinking beer, then that person is over 19 years old" is satisfied. This study builds upon earlier work by Johnson-Laird et al. (1972), demonstrating a similar phenomenon when the task involves postal regulations.

This is an instance of a general phenomenon in experimental psychology: the role of contextual effects. If factors previously thought to be irrelevant for performing a task turn out to correlate with differences in subject behavior, this can drive more refined analyses of the task under investigation. Many compelling examples of this phenomenon can be found in behavioral game theory. For example, in the ultimatum game pairs of subjects are provided with some quantity of money (e.g. a dollar). The first subject offers some percentage of it to the second. If the second subject accepts, the subjects receive the money in the proportions proposed by the first participant. If the second subject rejects the offer, however, neither participant receives any money. From the standpoint of a simple utility maximization model, it is in the interests of the first subject to offer the second one a vanishingly small percentage of the pot, and in the interests of the second subject to always accept. However, in practice, this is not how subjects perform. Güth et al. (1982) demonstrated both that subjects rarely offer the other participant a vanishingly small amount of the pot, and that when they do,

the second participant will reject the offer. This retributive behavior is further complicated by the fact that play changes with repeated opportunities, with subjects whose offers were accepted earlier becoming more daring and those whose offers were rejected becoming more conservative. Given that judgments of fairness and retribution seem to drive behavior on the ultimatum game, it is perhaps unsurprising that average performance varies across culture and is correlated with differences in social norms (Oosterbeek et al., 2004).

In a more complicated example, the centipede game, two players take turns choosing to either cash out a pot or pass it on to the next player. Payoffs are structured to ensure that if your opponent cashes the pot out, you receive less than if you had cashed it out on the previous move. If, however, the game goes through another iteration, and your opponent passes the pot back to you, you will receive more. So, it is in the interests of both players to pass the pot as long as possible, but distrust provides an incentive to cash the pot out whenever it is in one's possession. Since your expectations about your opponent's behavior determine whether you should pass the pot or not, higher-order reasoning about opponent's beliefs and intensions is necessary for rational game play. It turns out that subjects perform much better in settings involving intuitive everyday physical representations than in highly abstract settings, even when the two are logically equivalent (Hedden and Zhang, 2002; Flobbe et al., 2008; Meijering et al., 2010). This demonstrates that both one's theory of mind and one's familiarity with game circumstances are involved in computing a solution to the centipede game.[5]

Returning to the Wason selection task, the pertinent question is: what exactly is different between Wason's original setup and those involving underage drinking and postal regulations, and how should this difference affect our computational model? Johnson-Laird et al. (1972) and Griggs and Cox (1982) concluded that humans are better at logical reasoning in domains with which they are familiar. Since the original Wason task involves an abstract domain of letters and numbers, subjects are confused and fail to reason correctly. Cosmides (1989) and Cosmides and Tooby (1992) expand on these results and argue that they tell us something about cognitive architecture. In particular, Cosmides and Tooby conjecture that questions about postal regulations, drinking laws, etc. trigger a "cheater detection module." This module is hard wired to reason effectively, but in the domain-general case (when cheating may not be involved), we have no innate tendency to behave logically. Ultimately, this amounts to the claim that reasoning is simpler in cheater detection cases, i.e. readily mapped onto logical structure, but more complex in domain-general cases, i.e. its logical structure is not readily apparent.

[5]Ongoing empirical research on the centipede game can be found at http://www.ai.rug.nl/SocialCognition/

The most recent work on the logical analysis of the Wason selection task is a collaboration between psychologist Keith Stenning and logician Michiel van Lambalgen (2008). They point out that Wason's assertion that there is only one correct answer to the task is too quick, as it assumes a single interpretation of the experimental setup. Subjects who interpret the described rule as stating some other kind of dependency between D's and 3's than that captured by the material conditional are not necessarily making an error. The key here is in figuring out the relevant difference between versions of the task on which subjects perform in accordance with classical rules and versions (such as the original) on which they do not. Is it because the latter are abstract and the former concrete? Because the latter are unfamiliar and the former familiar? Because the latter are domain-general while the former involve cheater detection? Stenning and van Lambalgen's novel suggestion here is that the crucial difference is in whether the subject interprets the task as merely checking satisfaction of instances or as actually determining the truth of a rule. In the case of familiar deontic rules, their truth is not at issue, only whether or not they are being satisfied. The deontic nature of these rules means that turning cards over cannot falsify them (i.e. underage drinking is still wrong, even if one discovers that it occurs), and this strictly limits interpretation of the task to checking the rule has been satisfied. In contrast, the original version of the task may be interpreted as involving either a descriptive or a prescriptive rule, greatly increasing the cognitive burden on the subject.

3.2 The Frame Problem and Non-monotonic Logics

The Wason selection task provides a specific example where human behavior does not obey the rules of classical logic. As it turns out, human reasoning *must* violate the basic properties of classical logic in order to deal with an underspecified world. The problem here is not that classical logic is a poor reasoning strategy, but rather that it assumes a stronger relationship between evidence and circumstances than actually holds in most situations.

In a complex and changing world, humans receive limited evidence about the circumstances in which they find themselves. Crucially, this evidence is *defeasible*, i.e. the natural conclusions to draw from it may be defeated by later evidence. For example, suppose I wake up in a strange place and hear voices around me speaking in Chinese; I might conclude that I am in a Chinese restaurant. When I feel the surface on which I lie gently undulating, however, I might revise my conclusion, deciding instead that I have been Shanghaid, and am currently a passenger on a Chinese junk. Although my evidence has increased, my conclusions have changed. Modeling this type of reasoning requires a connective which is *non-monotonic*.

A function f is said to be *monotonic* if $n \leq m$ implies $f(n) \leq f(m)$; essentially, as the input grows, the output grows as well. Reasoning in

classical logic is monotonic because adding new premises always allows you to generate more conclusions. Let T and T' represent consistent sets of sentences and let $F(T)$ denote the deductive closure of T (i.e. the set of all sentences which follow from T by some specified (classical) inferential rules). Then, for all classical logics, $T \subseteq T'$ implies $F(T) \subseteq F(T')$.

Typically, a non-monotonic logic supplements an underlying classical logic with a new, non-monotonic connective and a set of inference rules which govern it. The rules describe a logic of *defeasible* inference, inferences which are usually safe, but which may be defeated by additional information. For example, from the fact that *this is a bird*, I can usually conclude that *this can fly*. This inference can be defeated, however, if I learn that *this is a penguin*. Symbolically, we want our system to ensure that $Bird(x) \Rightarrow Fly(x)$, but $Bird(x) \wedge Penguin(x) \not\Rightarrow Fly(x)$. Incidentally, this example also demonstrates why such a system is non-monotonic, since $\{Bird(x)\} \subset \{Bird(x), Penguin(x)\}$ yet $F(\{Bird(x)\}) \not\subset F(\{Bird(x), Penguin(x)\})$. Non-monotonic rules of inference go by a variety of names, including circumscription (McCarthy, 1980), negation as failure (Clark, 1978), and default reasoning (Reiter, 1980).

The original motivation for non-monotonic logic came from consideration of a particular type of defeasible reasoning, reasoning about a changing world. In a complex domain, humans are able to reason swiftly and effectively about both those features of the world which change *and those which do not* when some event occurs. The problem of how to keep track of those features of the world which do not change is called the "Frame Problem" (McCarthy and Hayes, 1969). The Frame Problem comes in both a narrow and a broad version (see discussion in Dennett, 1984). The broad version concerns the potential relevance of any piece of information in memory for effective inference. Philosophers of cognitive science have worried about this broad problem since at least Fodor (1983), and providing a solution may be equivalent to producing "human-level" A.I. The narrow problem, however, concerns keeping track of stability in a changing world. This problem is effectively solved by non-monotonic logic; for a complete history and detailed treatment, see Shanahan (1997).

The basic idea, however, is easy to see. If we allow ourselves default assumptions about the state of the world, we can easily reason about how it changes. For example, we might assume as default that facts about the world do not change unless they contradict incoming evidence. Learning that you ate eggs for breakfast does not change my belief that my tie is blue. Without the basic assumption that features of the world not mentioned by my incoming evidence do not change, I would waste all my computational resources checking irrelevant facts about the world whenever I received new information (e.g. checking the color of my tie after learning what you had for breakfast). This consideration inspired John McCarthy's assertion that, not only do "humans use ... 'non-monotonic' reasoning," but also "it is required

for intelligent behavior" (McCarthy, 1980, p. 28).

A more sophisticated form of default reasoning is found in systems which employ "negation as failure." Such a system may derive $\neg A$ provided it cannot derive A. Negation as failure is frequently implemented in systems using horn clauses, such as logic programming. Horn clauses state conditional relations such that the antecedent is a (possibly empty) conjunction of literals and the consequent is a single literal or *falsum*. For example, a program for reasoning about birds might contain the clause

$$f \longleftarrow b \wedge \neg p$$

where f stands for "fly," b for "bird," and p for "penguin." If it is part of the system's evidence that b, but it cannot derive p, then negation as failure will allow it to conclude f, as desired. In general, the semantics for systems involving negation as failure involve fixed points, e.g. finding the minimal model which satisfies all clauses (for a survey, see Fitting, 2002).

Although logic programming is a popular system for non-monotonic reasoning, it is not expressive enough to characterize all the possible varieties of non-monotonicity. Kraus et al. (1990) provide a unified approach to a hierarchy of non-monotonic logics of varying strengths. They distinguish each logic in this hierarchy in terms of the inference rules satisfied by their respective non-monotonic connectives, providing both a proof theory and a semantics. The basic strategy of their semantics for non-monotonic logics considers sets of worlds with an ordering relation defined on them. This ordering relation can be interpreted as preference or plausibility. $\alpha \Rightarrow \beta$ is true *iff* β is satisfied in all the most plausible worlds which satisfy α.

In order to get a flavor for the subtle differences between the systems considered by Kraus et al., consider the rule *Loop*:

$$\frac{\alpha_0 \Rightarrow \alpha_1, \alpha_1 \Rightarrow \alpha_2, \ldots, \alpha_{k-1} \Rightarrow \alpha_k, \alpha_k \Rightarrow \alpha_0}{\alpha_0 \Rightarrow \alpha_k} \qquad (Loop)$$

It should be obvious that a connective $\Rightarrow$ which satisfies *Loop* need not be as strong as the material conditional of classical logic. The material conditional satisfies transitivity ($A \rightarrow B$ and $B \rightarrow C$ imply $A \rightarrow C$), and *Loop* is an immediate consequence of transitivity (while the converse is not true). However, *Loop* is not satisfied in the weakest system considered by Kraus et al., which they call **C** for cumulative reasoning. The system **CL**, or cumulative reasoning with *Loop*, is an example of a non-monotonic reasoning system which is strictly weaker than classical logic, yet stronger than the weakest systems of non-monotonic reasoning.

Furthermore, there are additional systems in between **CL** and classical logic. Consider, for example, the rule

$$\frac{\alpha \Rightarrow \gamma, \quad \beta \Rightarrow \gamma}{\alpha \vee \beta \Rightarrow \gamma} \qquad (Or)$$

Or defines a system **P** strictly stronger than **CL**, but weaker than propositional logic. For example, *Loop* is a derived rule in **P**, yet contraposition ($\alpha \Rightarrow \beta$ implies $\neg\beta \Rightarrow \neg\alpha$) cannot be derived in **P**. Nevertheless, the distinction between **CL** and **P** is collapsed once we restrict attention to Horn clauses:

Theorem 3.1 (Kraus et al. (1990)) *If* M *is a set of horn clauses, then the horn clause p may be derived from* M *in the system* **P** iff *it may be derived from* M *in the system* **CL**.

Theorem 3.1 demonstrates that restricting attention to systems involving horn clauses (e.g. logic programming) collapses some of the subtle differences between possible systems for non-monotonic reasoning. Does this matter? If our interest is solving the frame problem, i.e. identifying a logical system for efficient reasoning about a changing world, then perhaps there is a single "correct" system to discover. This may even be an empirical question: which non-monotonic system do humans actually employ for "common sense" reasoning? However, the power of non-monotonic logics in cognitive science is not limited to computational level analysis. As we shall see in the next section, they are also powerful tools for analyzing the structure of the implementation level. For this purpose, we may wish to keep a full tool kit of non-monotonic systems of varying strengths available!

4 Between Algorithm and Implementation

How are computations performed in the brain? The answer which has dominated neuroscience since the late nineteenth century is the neuron hypothesis of Ramón y Cajal. Cajal was the first to observe and report the division of brain tissue into distinct cells: neurons. More importantly, he posited a flow of information from axon to dendrite through this web of neural connections, which he notated by drawing arrows on his illustrations of neural tissue. From the computational perspective, it is natural to identify this flow of information from neuron to neuron as the locus of the computation for solving cognitive tasks.

It is worth noting that this is not the only game in town. A plausible alternative to the neuron hypothesis comes from the dynamical systems perspective, which asserts that the behavior of a family of neurons cannot be reduced to signals communicated between them. Instead, this perspective asserts that computations should be modeled in terms of a dynamical system seeking basins of attraction. Perhaps the most strident defender of the dynamical systems perspective is Walter J. Freeman (1972, 2000).

Logic provides an abstract symbolic perspective on neural computation. As such, it can never be the whole story of the implementation level (which

by definition involves the physical instantiation of an algorithm). Nevertheless, logic can help bridge the gap between the implementation and algorithmic levels by analyzing structural similarities across different proposed instantiations. For example, if we subscribe to the neuron hypothesis, it is natural to look for logic gates in the wiring between neurons. But we may also look for logic gates in the wiring between families of neurons, or equivalent structure in the relations between basins of attraction in a dynamical system. Such an analysis can both shed light on the significant features of an implementation and resolve debates about features of the implementation level which are independent of its exact physical instantiation.

4.1 Implementation at the Neural Level

The classic work of Warren S. McCulloch and Walter H. Pitts (1943) proved the first representation theorem for a logic in an artificial neural network. In general, a representation theorem demonstrates that for every model of a theory, there exists an equivalent model within a distinguished subset. In this case, the "theory" is just a time stamped set of propositional formulas representing a logical derivation, and the distinguished subset in question is the set of neural networks satisfying a particular set of assumptions (e.g. neural firing is "all or none," the only delay is synaptic delay, the network does not change over time, etc.). McCulloch and Pitts show the opposite direction as well, i.e. the behavior of any network of the specified type can be represented by a sequence of time stamped propositional formulas. The propositions need to be time stamped to represent the evolution of the network through time; this allows a logical description of, say, how the activations of neurons at time t are constrained by the activations of neurons at time $t-1$.

McCulloch and Pitts had shown how neurons could be interpreted as performing logical calculations, and thus, how their behavior could be described and analyzed by logical tools. Furthermore, their approach was modular, as they demonstrated how different patterns of neural wiring could be interpreted as logic gates, signal junctions which compute the truth value of the conjunction, disjunction, or negation of incoming signals. Unfortunately, the applicability of their approach in neuroscience was limited by the plausibility of their assumptions. Although plausible at the time, the assumptions that neural behavior is all or nothing, or that networks remain unchanged over time, have turned out to be woefully inadequate for modeling real neural behavior.

Nevertheless, logical methods continue to provide insight into the structure of neural computation. In the face of an increasingly complex theory of neurophysiology, two obvious research projects present themselves. One suggests starting with a detailed look at the structure of individual neurons. Sandler and Tsitolovsky (2008), for example, begin with a detailed exam-

ination of the biological structure of the neuron, then develop a model of its behavior using fuzzy logic. The advantage of this approach is a close fit between formal model and biological reality, but the increase in detail necessitates a tradeoff against the insight which might be gained from a simpler, idealized model.

The second research strategy investigates the dynamics of artificial neural networks, looking for logical structure in their behavior. For example, Vogels and Abbott (2005) ran a number of simulations on large networks of integrate-and-fire neurons. Although not sensitive to every known detail of neurophysiology, the artificial neurons in these networks include many realistic features, such as a resting potential and a reset time after each action potential is generated. Although these artificial neurons are still clearly oversimplified idealizations compared to biological neurons, the hope is that enough relevant features have been captured to model the general shape of neural group dynamics realistically. After randomly generating such networks, Vogels and Abbott investigated their behavior to see if patterns of neurons exhibited the characteristics of logic gates. For example, they successfully identified patterns of activation corresponding to NOT, XOR, and other types of logic gate within their networks. To the extent that we endorse the idealizations made when running their simulations, we may take this as evidence that the algorithmic properties of neural computation may effectively be described in terms of logic gates.

This brings us to one of the many grey areas in Marr's breakdown of the computational hypothesis into three levels: are logic gates part of the algorithmic level, or the implementation level? Certainly, they do not represent the entire story of an implementation, since logic gates can be instantiated by many different physical structures. On the other hand, logic gates by themselves do not constitute an algorithm, which in general may be made up of many such logic gates arranged in an elaborate pattern. If one can reduce the relevant features of the implementing system to a description in terms of logic gates, however, one will at least have shown that any algorithm which itself can be broken down into logic gates is potentially a candidate for implementation in that system.

So, at the very least, the type of analysis discussed in this section demonstrates that concerns from the implementation level can constrain analysis at the algorithmic level. But if the same algorithm can be described at different degrees of abstraction (as a mathematical formula, a sequence of logic gates, a neural network, etc.), is there a correct level at which to explain an information processing task? Is there some *in principle* difference between an analysis offered in terms of neural networks and one offered in terms of logical rules? This was the subject of the symbolic / connectionist debate in the early 1990's, a debate which has been dissolved by a sequence of representation theorems more sophisticated than that first offered by McCulloch and Pitts.

4.2 The Symbolic / Connectionist debate

In an influential paper, Fodor and Pylyshyn (1988) argued that (i) mental representations exhibit systematicity; (ii) representations in neural networks do not exhibit systematicity; therefore (iii) the appropriate formalism for modeling cognition is symbolic (not connectionist). Systematicity here is just the claim that changes in the meaning of a representation correspond systematically to changes in its internal structure (e.g. from my ability to represent "John loves Mary," it follows that I can also represent "Mary loves John"). Fodor and Pylyshyn claim that the only case in which representations in a neural network do exhibit systematicity is when the network is a "mere" implementation of a symbolic system.[6]

It is important to notice what is at stake here: if cognitive tasks manipulate representations, then the appropriate analysis of a cognitive task must respect the properties of those representations. The claim that explanations in cognitive science must be in terms of symbolic systems does not, however, restrict attention to the computational level. Paradigmatic examples of the symbolic approach in cognitive science such as Chomsky (1957) investigate the role of particular algorithms for solving information processing tasks (such as extracting syntactic structure from a string of words). Nevertheless, the claim is that somewhere between abstract task specification and physical implementation, explanatory power breaks down, and neural networks fall on the implementation side of this barrier.

The response from connectionist modelers was violent and univocal: Fodor and Pylyshyn had simply misunderstood the representational properties of neural networks. Responses elucidated how representations in neural networks are "distributed" or "subsymbolic". Smolensky (1987, 1988, 1991), van Gelder (1990, 1991), Clark (1993), and many others all emphasized the importance of acknowledging the distinctive properties of distributed representations in understanding the difference between neural networks and symbolic systems. Yet it is difficult to put one's finger on what the essential feature of a distributed representation is which makes it qualitatively different from a symbolic representation. Since the late 1990's, the supposed distinction has largely been ignored as hybrid models have risen to prominence (e.g. the ACT-R architecture of Anderson and Lebiere, 1998, or the analysis of concepts in Gärdenfors, 2000). These hybrid models combine neural networks (for learning) and symbolic manipulation (for high-level problem solving). Although pragmatically satisfying, the hybrid approach avoids rather than resolves any questions about the essential difference between symbolic and distributed representations.

Is there some *in principle* difference between subsymbolic computation by neural networks over distributed representations and symbolic compu-

[6] Although they do not indicate how such implementational networks avoid their general critique, see Chalmers, 1990.

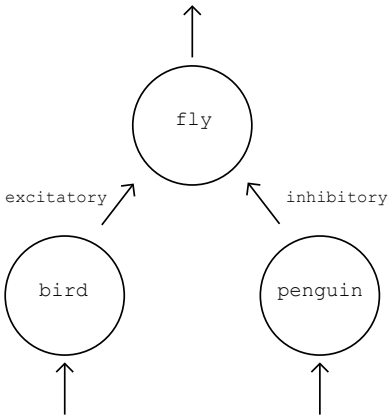

Figure 1: Neural network for non-monotonic reasoning about birds.

tation by Turing (or equivalent) machines? The representation theorem of McCulloch and Pitts suggests differently, namely that logical theories and neural networks are essentially the same, i.e. their computational and representational properties are equivalent. Can this result be extended to more realistic neural networks? As it turns out, a sequence of such results have been proven, demonstrating a deep conceptual equivalence between neural networks and non-monotonic logics.

It should be easy to see that some particular non-monotonic theories may be represented by neural networks. Consider the system discussed above for reasoning about birds: 2 input nodes (one for $Bird(x)$ and one for $Penguin(x)$) and an output node (for $Fly(x)$) are all we need to model this system with a simple neural network (Figure 1). So long as there is an excitatory connection from $Bird(x)$ to $Fly(x)$ and an even stronger inhibitory connection from $Penguin(x)$ to $Fly(x)$, this network will produce the same conclusions from the same premises as our non-monotonic theory. But this is just a specific case; a representation theorem for non-monotonic logics in neural networks would show us that for *every* non-monotonic theory, there is some neural network which computes the same conclusions. Such a theorem would demonstrate a deep computational equivalence between non-monotonic logics and neural networks.

As it turns out, representation theorems of this form have been given by several logicians coming from a variety of backgrounds and motivations. Hölldobler and collaborators prove a representation theorem for logic programs, demonstrating that for any logic program P, a three layer, feed forward network can be found which computes P (Hölldobler and Kalinke, 1994; Hitzler et al., 2004). Pinkas (1995) provides similar results for a wider class of neural networks and penalty logic. (Penalty logic is a non-monotonic logic which weights conditionals with positive integers representing the "penalty" if that conditional is violated. Reasoning in penalty

logic involves identifying the set of propositions which minimizes the overall penalty for a set of these weighted conditionals.) Hölldobler and Pinkas are both working within the artificial intelligence community and, consequently, their results are focussed on practical applications, with an emphasis on algorithms for performing the translation from symbolic to connectionist system (and, in the case of Pinkas, vice versa).

Stenning and van Lambalgen (2008) supplement their discussion of the Wason selection task with a proposal for a neurally plausible algorithm which generates behavior qualitatively similar to that experimentally observed in humans. Along the way, they prove a representation theorem for 3-valued logic programs in coupled neural networks. 3-valued logic programs can assign three distinct truth values to proposition letters: true, false, or "undecided." Here, undecided plays a role similar to negation as failure, though using three truth values allows for greater flexibility than two. Coupled neural networks are sheets of linked isomorphic networks, such that each node in the first network has a link to the corresponding node in the second network.

Theorem 4.1 (Stenning and van Lambalgen (2008)) *If* P *is a 3-valued logic program and* CN(P) *is the associated coupled neural network, then the least fixed point model of* P *corresponds to the activation of the output layer of* CN(P).

Unlike those of Hölldobler and Pinkas, Stenning and van Lambalgen's result concerns neural networks directly inspired by human neural architecture. To motivate their coupling model, they point to the extensive evidence for isomorphic mappings between layers of neurons in many parts of the human brain. Nevertheless, it is not clear that any of these mappings exhibit the logical structure postulated by Stenning and van Lambalgen, nor whether this is the appropriate level of neural detail at which to model behavior on the Wason selection task.

Since these results all concern varieties of logic programs, they collapse many of the fine-grained distinctions between non-monotonic systems discussed by Kraus et al. (1990). Inspired by the plurality of possible non-monotonic inference rules, Leitgeb (2001, 2003) proves a sequence of representation theorems for each system introduced by Kraus et al. in distinguished classes of neural networks. These results involve inhibition nets with different constraints on internal structure, where an inhibition net is a spreading activation neural network with binary (i.e. firing or non-firing) nodes and both excitatory and inhibitory connections. For example, if we assume there is no hierarchical structure to a net N, it exhibits non-monotonic reasoning as weak as **C**, but once we stipulate that N is hierarchical (i.e. contains no loops), it exhibits non-monotonic reasoning at least as strong as **CL**. The additional structure required to interpret the network as reasoning

in accordance with the rules of **P** is the constraint that for any formulas α and β, $\mathfrak{I}(\alpha \vee \beta) = \mathfrak{I}(\alpha) \cap \mathfrak{I}(\beta)$, where $\mathfrak{I}$ is the function which interprets formulas as sets of nodes in N.[7] Leitgeb (2005) extends these results from the particular case of inhibition nets to a much more general dynamical systems framework, although it appears that the close relationship between hierarchical structure and **CL** holds in this general case as well.

Unlike the other results discussed here, Leitgeb takes pains to ensure his representation theorems subsume the distributed case. In particular, the interpretation function $\mathfrak{I}$ may map a propositional formula to a set of nodes, i.e. distributing its representation throughout the network. From a philosophical standpoint, this result should raise questions for the debate between symbolic and connectionist approaches. Leitgeb has shown that any dynamical system performing calculations over distributed representations may be interpreted as a symbolic system performing non-monotonic reasoning. Correspondingly, it appears as if there is no substantive difference in representational or problem-solving power between symbolic and connectionist systems.

Of course, this is an "in principle" result; Leitgeb does not offer algorithms for constructing interpretations or extracting symbolic systems from trained neural nets. But the original argument from Fodor and Pylyshyn was an in principle argument as well. The representation theorems discussed here seem to show that no such in principle distinction can be found. If there is a distinction between symbolic systems and neural networks, it is not to be made at the level of what they can do or how they represent. Rather, the relevant distinctions seem to be wholly pragmatic. The advantages of parallel processing or neural plausibility may inspire a researcher to employ a neural network model rather than a symbolic system. For example, if the relevant features of the data necessary for completing a task successfully are not known.[8] Likewise, when analytical rules can more easily be found, this motivates working within a symbolic paradigm (e.g. the Chomskian program in linguistics). This inspires the question of whether there is an in principle distinction between neural networks and symbolic systems *at the level of computational efficiency*.

[7] Obviously, this glosses over many of the specifics of Leitgeb's account, e.g. how to handle negation. See Leitgeb (2003) for the complete details.

[8] See, for example, the famous mine / rock differentiating neural network due to Gorman and Sejnowski (1988). They trained a neural network to distinguish between sonar signals taken from a metal cylinder and those taken from a cylindrical rock, producing a success rate comparable to that of human sonar operators. The motivations for the approach were two-fold: first, the computational efficiency of parallel processing over traditional non-parametric methods; second, the flexibility with which inputs could be specified, since a complete prior determination of the features relevant for discrimination was unnecessary.

5 Computational Complexity Theory

The Church-Turing Thesis provides a distinction between those problems which are computable and those which, in general, are not. Complexity theory supplements the computational perspective with more fine-grained distinctions, in particular, a distinction between those problems which can be solved *efficiently* and those which cannot. Efficiency considerations can help to bridge the gap between computational and algorithmic levels of analysis. Before looking at some specific examples, however, we first introduce the basic complexity classes and consider the extent to which the structure of a machine (e.g. a Turing machine vs. a neural network) affects complexity considerations.

5.1 P vs. NP

Some problems, although computable, nevertheless require too much time or memory to be feasibly solved by a realistic computational device. Computational complexity theory investigates the resources (time, memory, etc.) required for the execution of algorithms and the inherent difficulty of computational problems (see e.g. Papadimitriou, 1993; Arora and Barak, 2009, or the paper by Ramanujam in this volume). This means that the theory does not deal directly with concrete algorithmic procedures, but instead studies the abstract computational properties of queries. Given a problem, features which hold for all possible solution algorithms can be investigated in a precise mathematical sense. This allows us to precisely distinguish those problems which have efficient solutions from those which do not.

This method for analyzing queries allows us to sort them into complexity classes. In particular, we want to identify efficiently solvable problems and draw a line between tractability and intractability. In general, the most important distinction is that between problems which can be computed in polynomial time with respect to the size of the problem, i.e. relatively quickly, and those which are believed to have only exponential time algorithmic solutions. The class of problems of the first type is called PTIME (P for short); one can demonstrate that a problem belongs to this class if one can show that it can be computed by a deterministic Turing machine in polynomial time. Problems belonging to the second class are referred to as NP-hard. Intuitively, a problem is NP-hard if there is no efficient algorithm for solving it. The only way to deal with it is by using brute-force methods: searching through all possible combinations of elements over a universe. Importantly, contrary to common suggestions in the cognitive science literature (see e.g. Chater et al., 2006) computational complexity theory has shown that many intractable (e.g. NP-hard) functions cannot be efficiently approximated (see e.g. Ausiello et al., 2000). In other words, NP-hard problems lead to combinatorial explosion.

Notice that this categorization is helpful only under the assumption that the complexity classes defined in the theory are essentially different. These inequalities are usually extremely difficult to prove. In fact, the most famous problem in complexity theory is of this form, namely the widespread assumption that P≠NP. This is considered one of the seven most important open mathematical problems by the Clay Institute of Mathematics, who have offered a $1,000,000 prize for its solution. Speaking precisely, NP-hard problems are problems which are at least as difficult as problems belonging to the NPTIME (NP) class; this is the class of problems which can be computed by nondeterministic Turing machines in polynomial time. NP-complete problems are NP-hard problems belonging to NPTIME, hence they are intuitively the most difficult problems among the NPTIME problems. If we could show that any NP-complete problem is PTIME computable, we would have demonstrated that P=NP. Whether this is possible or not is still an open question. Nevertheless, computer science generally (and computational complexity in particular) operates under the assumption that these two classes are different, an assumption which has proved enormously fruitful in practice.

Before we move to more general considerations, let us consider an example. Many natural problems are computable in polynomial time, for instance calculating the greatest common divisor of two numbers or looking something up in a dictionary. However, we will focus here on a very important NP-complete problem, the satisfiability problem for classical propositional logic (SAT). The problem is to decide whether a given classical propositional formula is not a contradiction. Let φ be a propositional formula with $p_1, \ldots, p_n$ distinct variables. One well-known algorithm simply checks the truth-table to see if φ has a satisfying valuation. How big is the truth-table for φ? The formula has n distinct variables occurring in it and therefore the truth-table has 2^n rows. If $n = 10$ there are 1,024 rows, for $n = 20$ there are already 1,048,576 rows and so on. In the worst case, to decide whether φ is satisfiable we have to check all rows. So, the time needed for this familiar algorithm to find a solution is exponential with respect to the number of different propositional letters of the formula. A seminal result of computational complexity theory states that this is not a property of the truth-table method but of the inherent complexity of the satisfiability problem. We have the following:

Theorem 5.1 (Cook (1971)) SAT *is NP-complete.*

The previous paragraph tells us something about the relation between classical logic and computability theory: even very simple logics can define extremely difficult computational problems. Classical descriptive complexity theory deals with the relationship between logical definability and computational complexity. The main idea is to treat classes of finite models over a fixed vocabulary as computational problems. In such a setting rather than

the computational complexity of a given class of models we are dealing with its descriptive complexity, i.e. what strength of logical language is needed to define the class. The seminal result here is Fagin's theorem establishing a correspondence between existential second order logic and NP:

Theorem 5.2 (Fagin (1974)) Σ_1^1 *captures* NP.

It says that for every property φ, it is definable in the existential fragment of second-order logic, Σ_1^1, if and only if it can be recognized in polynomial time by a non-deterministic Turing machine (see Immerman, 1999).

How about non-classical logics; in particular, what do we know about the computational complexity of reasoning with non-monotonic logics? It turns out that typically the computational complexity of non-monotonic inferences is higher than the complexity of the underlying monotone logic. As an example, restricting the expressiveness of the language to Horn clauses allows for polynomial inference as far as classical propositional logic is concerned. However, this inference task becomes NP-hard when propositional default logic or circumscription is employed. This increase in complexity comes from the fixed-point constructions needed to provide the semantics for negation as failure and other non-monotonic reasoning rules. In general, determining minimality is intractable (see Cadoli and Schaerf, 1993, for a survey).

5.2 General Tractability Borders

In the early days of computational complexity theory, the following thesis was formulated independently by Alan Cobham (1965) and Jack Edmonds (1965):

The class of practically computable problems is identical to the PTIME class, that is, the class of problems which can be computed by a deterministic Turing machine in a number of steps bounded by a polynomial function of the length of a query.

This thesis is accepted by most computer scientists. For example, Garey and Johnson (1979) identify discovery of a PTIME algorithm with producing a real solution to a problem:

> 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.

The common belief in the Cobham-Edmonds Thesis stems from the practice of programmers. NP-hard problems often lead to algorithms which are not practically implementable even for inputs of not very large size. Assuming the Church-Turing Thesis and $P \neq NP$, we come to the conclusion that this has to be due to some internal properties of these problems and not to the details of current computing technology. However, even with these assumptions, there are still some doubts. Examining these worries will lead to a better understanding of the nature of claims about computational complexity.

First of all, in some cases the polynomial algorithm is essentially better just for very large data. Consider, for example, an algorithm bounded by $n^{\frac{1}{5}\log\log n}$. Such an algorithm could be used practically even though it is not polynomial. The reason is very simple: $n^{\frac{1}{5}\log\log n} > n^2$ only when $n > e^{e^{10}}$, where $e \approx 2.718281$ (Gurevich, 1993). In other words, the polynomial algorithm is essentially better only for very big input. An example of such an algorithm is the Adleman, Pomerance, and Rumely deterministic algorithm for checking whether a given number n is prime, which has a bound of the form $n^{c\log\log n}$, where c is a positive constant (Adleman et al., 1983).[9]

Second, a polynomial algorithm might also be practically intractable. $n^{98466506514687}$ is a polynomial, but even for small n an algorithm of that working time would not be practical. However, many theorists believe that if we come up with polynomial algorithms of a high degree then it is only a matter of time before they will be simplified. In practice, programmers implement mainly algorithms with time complexity not greater than n^3. On the other hand, exponential procedures are sometimes used if they are supposed to work on small input data. Related additional difficulty comes from the fact that computational complexity measures do not take constants into considerations as when n increases their influence on the complexity decreases. For example, an algorithm working in time $n^3 + 2^{630}$ is still taken as a reasonable polynomial bound of degree 3.

Finally, let us consider an even more drastic example (see Gurevich, 1995). Let g be an uncomputable function and define f as follows:

$$f(x) = \begin{cases} 1 & \text{if length of } x < 2^{9999999999999999999999999999999} \\ g(x) & \text{otherwise.} \end{cases}$$

Is f computable? For all inputs of reasonable size f is computable and has value 1. It is uncomputable only for extremely big inputs. Therefore, can we intuitively claim that f is computable?

This discussion shows that the Cobham-Edmonds Thesis only provides an approximation of the tractability / intractability boundary. Moreover,

[9] In 2002, Indian scientists at IIT Kanpur discovered a new deterministic algorithm known as the AKS algorithm. This algorithm checks whether a number n is prime in an amount of time which is a polynomial function of the logarithm of n, demonstrating that primality is in PTIME (Agrawal et al., 2004).

for the claim to make sense, we need some additional assumptions which will return us to the symbolic / connectionist debate.

5.3 The Invariance Thesis

Computational complexity theory is concerned with the inherent complexity of problems independent of particular algorithmic solutions and their implementations. However, this analysis will be of little interest if it is not also independent of the particular computational device on which the algorithm is implemented. The most common model of computation used in complexity analysis is the Turing machine, yet Turing machines have a radically different structure from that of modern digital computers, and even more so from that of neural networks. In order to justify the application of results from complexity theory (e.g. that a particular problem is intractable) in cognitive science, we need to demonstrate that they hold independent of any particular implementation.

The Invariance Thesis (see e.g. van Emde Boas, 1990) states that:

Given a "reasonable encoding" of the input and two "reasonable machines," the complexity of the computation performed by these machines on that input will differ by at most a polynomial amount.

By "reasonable machine," we mean any computing device which may be realistically implemented in the physical world. The situation here is very similar to that of the Church-Turing Thesis: although we cannot prove the Invariance Thesis, the fact that it holds for all known physically implementable computational devices provides powerful support for it. Of course, there are well-known machines which are ruled out by the physical realizability criterion; for example, non-deterministic Turing machines and arbitrarily precise analog neural networks are not realistic in this sense. Assuming the Invariance Thesis, we get that a task is difficult if it corresponds to a function of high computational complexity, independent of the computational device under consideration, at least as long as it is reasonable.

It is worth discussing in a little more detail why neural networks fall within the scope of the Invariance Thesis. Neural networks can provide a speed-up over traditional computers because they can perform computations in parallel. However, from the standpoint of complexity theory, the difference between serial and parallel computation is irrelevant for tractability considerations. The essential point is this: any realistic parallel computing device will only have a finite number of parallel channels for simultaneous computation. Since this number is finite, it will not provide an in principle difference in computational power, but only a polynomial amount speed-up over a similar serial device. In particular, if the parallel device has n channels, then it should speed up computation by a factor of n (providing it can

use its parallel channels with maximum efficiency). As the size of the input grows significantly larger than the number of parallel channels, the advantage in computational power for the parallel machine becomes less and less significant. For example, the polynomial speed-up of parallel computation provides a vanishingly small advantage on NP-hard problems where the solution time grows exponentially. Therefore, the difference between symbolic and connectionist computations is negligible from the tractability perspective (see van Rooij, 2008, particularly Section 6.6, for extended discussion).

These considerations should clarify and mitigate the significance of the representation theorems discussed in Section 4.2. How can we reconcile these representations with the observation in Section 5.1 that fixed point constructions are NP-hard? The answer is that these representation theorems generally show that for a non-monotonic theory *with n proposition letters*, there exists a neural network which computes it with m nodes, where $m \geq n$. So, it is not that an NP-hard problem (finding a fixed point) is in general computed efficiently, but rather, the problem as bounded by the size of the propositional variables under consideration can be computed by a device *with a sufficient number of parallel channels*.

6 Beyond Algorithms: Efficiency and Cognition

If we accept the considerations in the previous section, then conclusions from complexity analysis are invariant with respect to the details of the particular implementation of a cognitive device. Consequently, we can use computational complexity theory to investigate the grey area between Marr's computational and algorithmic levels. Since cognitive systems are physical systems, they perform tasks under computational resource constraints. Therefore, the functions computed by cognitive systems need to be computable in realistic time and with the use of a realistic amount of memory. For example, we may posit that task analysis at the computational level be constrained to tasks which have PTIME solutions. More generally, if behavior on a task slows with input size in the proportion suggested by complexity analysis, we may take this as evidence that we have correctly identified the task being performed.

6.1 The P-Cognition Thesis

The worry that plausible models of agents in a complex world must take into account the limits of their computational resources is often called the problem of bounded rationality (Simon, 1957, and many following publications). Simon argued that the limited computational resources of bounded agents require them to solve many problems with rough heuristics rather than exhaustive analyses of the problem space. In essence, rather than solve the hard problem presented to him by the environment, the agent solves

an easier, more tractable problem which nevertheless produces an acceptable solution. In order to apply this insight in cognitive science, it would be helpful to have a precise characterization of which class of problems can plausibly be computed by the agent. The answer suggested by complexity theory is to adapt the Cobham-Edmonds Thesis:

The class of problems which can be computed by a cognitive agent is approximated by the PTIME class, i.e. bounded agents can only solve problems with polynomial time solutions.

As far as we are aware, the version of the Cobham-Edmonds Thesis for cognitive science was first formulated explicitly in print by Frixione (2001) and later dubbed the P-Cognition Thesis by van Rooij (2008). The P-Cognition Thesis states that a cognitive task is easy (hard) if it corresponds to a(n) (in)tractable problem.

The P-Cognition Thesis can be used to analyze which problem an agent is plausibly solving when the world presents him with an (apparently) intractable problem. For example, Levesque (1988) argues that the computational complexity of general logic problems motivates the use of Horn clauses and other tractable formalisms to obtain psychologically realistic models of human reasoning. Similarly, Tsotsos (1990) emphasizes that visual search in its general (bottom-up) form is NP-complete. As a consequence, only visual models in which top-down information constrains visual search space are computationally plausible. In the study of categorization and subset choice, computational complexity serves as a good evaluation of psychological models (see e.g. van Rooij et al., 2005). This general strategy for analyzing cognitive tasks can also be found in philosophy of mind; for example, Cherniak (1981) argues that tractability considerations demand a philosophical analysis of the conditions required for a minimal notion of rationality. Is there any empirical evidence for the psychological plausibility of the P-Cognition Thesis?

6.2 Inefficiency and Task Analysis

In Section 5.1 we saw how descriptive complexity can be used to analyze formal languages, but what about natural language? Some of the earliest research trying to combine computational complexity with semantics can be found in Sven Ristad's book "The Language Complexity Game" (1993). Ristad carefully analyzes the comprehension of anaphoric dependencies in discourse. He considers a few approaches to describing the meaning of anaphora and proves their complexity. Finally, he concludes that the problem is inherently NP-complete and that all good formalisms accounting for it should be exactly as strong.

More recently, Pratt-Hartmann (2004) showed that different fragments

of natural language capture various complexity classes. More precisely, he studied the computational complexity of satisfiability problems for various fragments of natural language. Pratt-Hartmann proved that the satisfiability problem for the syllogistic fragment is in PTIME, as opposed to the fragment containing relative clauses, which is NP-complete. He also described fragments of language such that their computational complexity is even harder (with non-copula verbs or restricted anaphora) and finally he provides the reader with an undecidable fragment containing unrestricted anaphora.

This work appears to challenge the P-Cognition Thesis—if satisfiability is NP-complete, or even uncomputable, for fragments of natural language, it appears that we are forced, not only to abandon the P-Cognition Thesis, but even to abandon the Church-Turing Thesis, and posit that cognition involves super-Turing computation. As discussed above, this conclusion contradicts all available evidence on physical systems. More importantly, however, it constitutes a dead-end for empirical research: if the brain is super-Turing, then all bets are off as to how it behaves on any task. The P-Cognition Thesis provides a constructive way to move forward. The negative results of Ristad and Pratt-Hartmann demonstrate that the brain is not solving the complete satisfiability problem when interpreting sentences of natural language. This motivates the search for polynomial time heuristics which might plausibly compute interpretations of sentences of natural language.

For example, Pagin (2009) tries to explain compositionality in terms of computational complexity, cognitive difficulty as experienced by language users during communication, and language learnability. He argues that compositionality simplifies the complexity of language communication. Similarly, the second author of this paper has applied computational complexity analysis to the psycholinguistic study of the meaning of quantifiers in natural language (Szymanik, 2007, 2009). The idea is that the cognitive difficulty of quantifier processing can be assessed on the basis of the complexity of the corresponding minimal automata (van Benthem, 1986) required to compute their meaning. Szymanik and Zajenkowski (2010a,b) investigated this hypothesis in empirical studies by comparing the processing of various classes of quantifiers with respect to their computational complexity. The authors concluded that subjects' reaction time and working memory involvement increased as predicted by the complexity analysis. Just to give one example, from a theoretical perspective, most of the natural language quantifiers, like 'some', 'all' (Aristotelian), 'more than 5', 'less than 7' (cardinal), and 'an even/odd number of' (parity) are recognizable by finite-state automata. However, proportional quantifiers ('most', 'less than half') require a stronger recognition mechanism with unbounded internal memory. In order to evaluate these quantifiers, the sizes of two sets need to be compared; this procedure may be simulated by a push-down automaton, but not a finite automaton. For instance, in order to verify the sentence "More than

half of the cars are red", one has to count and hold in short-term memory the number of red cars and then compare it with the total number of cars (cf. Pietroski et al., 2009). No such memorization / comparison is necessary when processing other quantifiers. Therefore, when compared across different classes of quantifiers, subjects should find proportional constructions to be the hardest to verify, as they put the highest demands on working memory. And indeed, comparatively slower performance when evaluating proportional quantifiers has been experimentally confirmed by Szymanik and Zajenkowski.

Finally, Szymanik (2010) studies the computational complexity of multi-quantifier sentences. He demonstrates the computational dichotomy between different readings of reciprocal sentences, for example:

1. Most of the parliament members refer indirectly to each other.

2. Boston pitchers were sitting alongside each other.

While the first sentence is usually given an intractable NP-hard reading the second one is interpreted by a PTIME computable formula. This motivates the conjecture that listeners are more likely to assign readings which are simpler to compute. The psychological plausibility of this conjecture is still awaiting further investigation; however, there are already some interesting early results (Bott et al., 2011).

These results demonstrate how the P-Cognition Thesis can drive experimental practice. Differences in performance (e.g. reaction time) can support a computational analysis of the task being performed (Szymanik and Zajenkowski, 2010a). Furthermore, one can track the changes in heuristic a single agent employs as the problem space changes (Bott et al., 2011). These techniques apply equally in domains other than language. For example, it is known that reaction time increases linearly when subjects are asked to count between 4 and 15 objects. Up to 3 or 4 objects the answer is immediate, so-called subitizing. For judgments involving more than 15 objects, subjects start to approximate: reaction time is constant and the number of incorrect answers increases dramatically (Dehaene, 1999).

Analogously, we can ask what gross differences in reaction time on NP-hard tasks say about the nature of the heuristics subjects employ. A particularly telling example here is the case of chess experts. The radical increase in reaction time of chess experts over that of novices does not indicate that they are solving the same (NP-hard) problem of exhaustively searching the tree of possible moves, but rather that they are solving the different (and much simpler) problem of searching only the "good" moves available to them (Simon and Simon, 1962). The P-Cognition Thesis guides our analysis of expert behavior here, suggesting that the algorithms experts implement for constraining search may be PTIME in complexity.

6.3 Objections and Future Directions

Computational complexity is defined in terms of limit behavior. It answers the question: as the size of the input increases *indefinitely*, how do the running time and memory requirements of the algorithm change? So, although computational complexity theory investigates the scalability of computational problems and algorithms, i.e. it measures the rate of increase in computational resources required as a problem grows, it may not apply to computations with fixed or bounded input size.

Nevertheless, computational complexity can still be reasonably interpreted as saying something about problem difficulty on a fixed model. Namely, if the computational complexity of the problem is high, then it means that there are no "clever" algorithms for solving it, i.e. we must perform an exhaustive search through the entire solution space. The pertinent question here for cognitive science is whether empirical or statistical considerations can bound plausible input size closely enough for the parallelism of the brain's computational resources to make NP-hard problems tractable. Certainly, no *in principle* bound can be provided. Furthermore, the results of the previous section indicate that there are many domains (language processing, counting) where tractability considerations fruitfully predict behavior.

A similar response can be made to the critic who questions the value of worst-case computational complexity as a measure of difficulty for cognitive processes. For example, it might be the case that inputs generated by nature have some special properties which make problems tractable on those inputs even though in principle they might be NP-hard. If so, we should turn our attention to average-case complexity theory, which studies the computational complexity of problems on randomly generated inputs. This theory is motivated by the fact that certain NP-hard problems are in fact easily computable on "most" of the inputs. But average-case complexity theory extends and supplements the worst-case analysis; it does not, in general, replace it. For example, if the appropriate probability distribution over nature's behavior is unavailable, average-case complexity analysis simply cannot be applied. Furthermore, it seems that the worst-case analysis is actually the right perspective from which to analyze cognitive tractability. Average-case complexity may still be preferred for purposes other than assessing tractability, e.g. comparing the time-complexity of different (tractable) algorithmic-level explanations with reaction time data obtained via experimentation (see van Rooij, 2008).

Another strategy for cognitively plausible complexity measures is to break up each task into parameters and analyze how each of the parameters contribute to the overall complexity. It might be the case that intractability of some problems comes from a parameter which is usually very small no matter how big the input (see van Rooij and Wareham, 2007, for ex-

amples). This way of thinking leads to parametrized complexity theory as a measure for the complexity of cognitive processes. Iris van Rooij (2008) investigates this subject, proposing the Fixed-Parameter Tractability Thesis as a refinement of the P-Cognition Thesis.

Bounds on input size, average-time complexity, and parameterized complexity do not replace traditional complexity theory, rather they supplement and elaborate upon it. All three of these alternatives to the P-Cognition Thesis require empirical support to be adequately applied. Conversely, the P-Cognition Thesis provides a constructive strategy for moving forward in the absence of these empirical details.

7 Conclusion

We've examined a number of applications of logical methods in cognitive science. If we accept the computational perspective, then no area of cognitive behavior is immune to logical analysis. Logic provides a particularly powerful tool, however, in the cracks between Marr's three levels. Representation theorems can probe an intermediate level between algorithms and physical implementations. Likewise, complexity theory can probe the relationship between computational task analysis and algorithmic solutions.

Throughout, we have focussed on *in principle* distinctions. In some cases these seem to provide a productive source for empirical predictions (the tractable / untractable distinction), in others they simply seem to obscure conceptual issues (the supposed distinction between symbolic and distributed representations). As our understanding of the brain and psychological behavior increases, the role of such distinctions will no doubt diminish in the face of specific models. Nevertheless, as long as we endorse the computational perspective, even these new models will be susceptible to new in principle analyses at the hands of logic.

Acknowledgments

We would like to thank Johan van Benthem, Iris van Rooij, and Keith Stenning for many comments and suggestions. The first author would also like to thank Douwe Kiela and Thomas Icard for many helpful discussions of this material. The second author was supported by NWO Vici grant 277-80-001.

References

Adleman, L. M., Pomerance, C., and Rumely, R. S. (1983). On distinguishing prime numbers from composite numbers. *Annals of Mathematics*, 117:173–206.

Agrawal, M., Kayal, N., and Saxena, N. (2004). Primes in P. *Annals of Mathematics*, 160(2):781–793.

Anderson, J. R. and Lebiere, C. (1998). *The Atomic Components of Thought*. Erlbaum, Mahwah, NJ.

Arora, S. and Barak, B. (2009). *Computational Complexity: A Modern Approach*. Cambridge University Press, 1 edition.

Ausiello, G., Crescenzi, P., Kann, V., Marchetti-Sp, Gambosi, G., and Spaccamela, A. M. (2000). *Complexity and Approximation: Combinatorial Optimization Problems and Their Approximability Properties*. Springer.

Benacerraf, P. (1967). God, the devil, and Gödel. *The Monist*, 51:9–32.

van Benthem, J. (1986). *Essays in logical semantics*. Reidel.

Bott, O., Schlotterbeck, F., and Szymanik, J. (2011). Interpreting tractable versus intractable reciprocal sentences. In Bos, J. and Pulman, S., editors, *Proceedings of the International Conference on Computational Semantics 9*, pages 75–84.

Cadoli, M. and Schaerf, M. (1993). A survey of complexity results for non-monotonic logics. *J. Log. Program.*, 17(2/3&4):127–160.

Chalmers, D. (1990). Why Fodor and Pylyshyn were wrong: The simplest refutation. In *Proceedings of the Twelfth Annual Conference of the Cognitive Science Society*, pages 340–347.

Chater, N., Tenenbaum, J., and Yuille, A. (2006). Probabilistic models of cognition: where next? *TRENDS in Cognitive Sciences*, 10(7):292–293.

Cheng, P. W., Holyoak, K. J., Nisbett, R. E., and Oliver, L. M. (1986). Pragmatic vs. syntactic approaches to training deductive reasoning. *Cognitive Psychology*, 18:293–328.

Cherniak, C. (1981). Minimal rationality. *Mind*, 90(358):161–183.

Chomsky, N. (1957). *Syntactic Structures*. Mouton de Gruyter.

Clark, A. (1993). *Associative Engines*. Bradford Books, Cambridge, MA.

Clark, K. L. (1978). Negation as failure. In Gallaire, H. and Minker, J., editors, *Logics and Data Bases*. Plenum Press, New York, NY.

Cobham, J. (1965). The intrinsic computational difficulty of functions. In *Proceedings of the 1964 International Congress for Logic, Methodology, and Philosophy of Science*, pages 24–30. North-Holland.

Cook, S. A. (1971). The complexity of theorem-proving procedures. In *STOC '71: Proceedings of the third annual ACM symposium on Theory of Computing*, pages 151–158, New York, NY, USA. ACM Press.

Cooper, B. S. (2003). *Computability Theory*. Chapman Hall/Crc Mathematics Series. Chapman & Hall/CRC.

Cosmides, L. (1989). The logic of social exchange: Has natural selection shaped how humans reason? Studies with the Wason selection task. *Cognition*, 31:187–276.

Cosmides, L. and Tooby, J. (1992). Cognitive adaptations for social exchange. In Barkow, J., Cosmides, L., and Tooby, J., editors, *The adapted mind: evolutionary psychology and the generation of culture*, pages 163–228. Oxford UP, Oxford, UK.

Dehaene, S. (1999). *The Number Sense: How the Mind Creates Mathematics*. Oxford University Press, USA.

Dennett, D. C. (1984). Cognitive wheels: The frame problem of AI. In Hookway, C., editor, *Minds, Machines and Evolution: Philosophical Studies*. Cambridge UP, Cambridge.

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

Fagin, R. (1974). Generalized first-order spectra an polynomial time recognizable sets. In Karp, R., editor, *Complexity of Computation*, volume 7 of *SIAM—AMS Proceedings*, pages 43–73. American Mathematical Society.

Feferman, S. (1995). Penrose's Gödelian argument. online publication, available at http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.130.7027&rep=rep1&type=pdf.

Fitting, M. (2002). Fixpoint semantics for logic programming a survey. *Theoretical Computer Science*, 278(1-2):25 – 51.

Flobbe, L., Verbrugge, R., Hendriks, P., and Krämer, I. (2008). Children's application of theory of mind in reasoning and language. *Journal of Logic, Language and Information*, 17(4):417–442.

Fodor, J. A. (1983). *The Modularity of Mind: An Essay on Faculty Psychology*. MIT Press, Cambridge, MA.

Fodor, J. A. and Pylyshyn, Z. W. (1988). Connectionism and cognitive architecture: a critical analysis. *Cognition*, 28:3–71.

Freeman, W. J. (1972). Waves, pulses and the theory of neural masses. *Progress in Theoretical Biology*, 2:87–165.

Freeman, W. J. (2000). *How Brains Make Up Their Minds*. Columbia University Press.

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

Gärdenfors, P. (2000). *Conceptual Spaces: The Geometry of Thought*. MIT Press, Cambridge, MA.

Garey, M. R. and Johnson, D. S. (1979). *Computers and Intractability*. W. H. Freeman and Co., San Francisco.

Gold, E. M. (1967). Language identification in the limit. *Information and Control*, 10:447–474.

Gorman, R. P. and Sejnowski, T. J. (1988). Analysis of hidden units in a layered network trained to classify sonar targets. *Neural Networks*, 1:75–89.

Griggs, R. A. and Cox, J. R. (1982). The elusive thematic-materials effect in Wason's selection task. *British Journal of Psychology*, 73:407–420.

Gurevich, Y. (1993). Feasible functions. *London Mathematical Society Newsletter*, 10(206):6–7.

Gurevich, Y. (1995). The Value, if any, of Decidability. *Bulletin of European Association for Theoretical Computer Science*, pages 129–135.

Güth, W., Schmittberger, R., and Schwarze, B. (1982). An experimental analysis of ultimatum bargaining. *Journal of Economic Behavior and Organization*, 3:367–388.

Hedden, T. and Zhang, J. (August 2002). What do you think I think you think?: Strategic reasoning in matrix games. *Cognition*, 85:1–36(36).

Hitzler, P., Hölldobler, S., and Seda, A. K. (2004). Logic programs and connectionist networks. *Journal of Applied Logic*, 2(3):245–272. Neural-symbolic Systems.

Hölldobler, S. and Kalinke, Y. (1994). Toward a new massively parallel computational model for logic programming. In *Proc. Workshop on Combining Symbolic and Connectionist Processing, ECAI-94*, Amsterdam.

Immerman, N. (1999). *Descriptive complexity*. Springer Verlag.

Johnson-Laird, P. N., Legrenzi, P., and Legrenzi, M. S. (1972). Reasoning and a sense of reality. *British Journal of Psychology*, 63:395–400.

Kraus, S., Lehmann, D. J., and Magidor, M. (1990). Nonmonotonic reasoning, preferential models and cumulative logics. *Artificial Intelligence*, 44(1–2):167–207.

Kugel, P. (1986). Thinking may be more than computing. *Cognition*, 22(2):137–198.

Leitgeb, H. (2001). Nonmonotonic reasoning by inhibition nets. *Artificial Intelligence*, 128(1–2):161–201.

Leitgeb, H. (2003). Nonmonotonic reasoning by inhibition nets II. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, 11(2 (Supplement)):105–135.

Leitgeb, H. (2005). Interpreted dynamical systems and qualitative laws: From neural networks to evolutionary systems. *Synthese*, 146:189–202.

Levesque, H. J. (1988). Logic and the complexity of reasoning. *Journal of Philosophical Logic*, 17(4):355–389.

Lucas, J. R. (1961). Minds, machines and Gödel. *Philosophy*, 36(137):112–127.

Marr, D. (1983). *Vision: A Computational Investigation into the Human Representation and Processing Visual Information*. W.H. Freeman, San Francisco.

McCarthy, J. (1980). Circumscription—a form of non-monotonic reasoning. *Artificial Intelligence*, 13(1–2):27–39.

McCarthy, J. and Hayes, P. J. (1969). Some philosophical problems from the standpoint of artificial intelligence. In Michie, D. and Meltzer, B., editors, *Machine Intelligence*, volume 4, pages 463–502. Edinburgh University Press.

McCulloch, W. S. and Pitts, W. H. (1943). A logical calculus immanent in nervous activity. *Bulletin of Mathematical Biophysics*, 5:115–133.

Meijering, B., Van Maanen, L., Van Rijn, H., and Verbrugge, R. (2010). The facilitative effect of context on second-order social reasoning. In Catrambone, R. and Ohlsson, S., editors, *Proceedings of the 32nd Annual Conference of the Cognitive Science Society*, pages 1423–1428, Austin (TX). Cognitive Science Society.

Oosterbeek, H., Sloof, R., and van de Kuilen, G. (2004). Cultural differences in ultimatum game experiments: Evidence from a meta-analysis. *Experimental Economics*, 7:171–188.

Pagin, P. (2009). Communication and the complexity of semantics. In Hinzen, W., Machery, E., and Werning, M., editors, *Oxford Handbook of Compositionality*. Forthcoming; Manuscript available at http://people.su.se/~ppagin/pagineng.htm.

Papadimitriou, C. H. (1993). *Computational Complexity*. Addison Wesley.

Penrose, R. (1994). *Shadows of the Mind : A Search for the Missing Science of Consciousness*. Oxford University Press, USA.

Pietroski, P., Lidz, J., Hunter, T., and Halberda, J. (2009). The meaning of 'most': semantics, numerosity, and psychology. *Mind and Language*, 24(5):554–585.

Pinkas, G. (1995). Reasoning, nonmonotonicity and learning in connectionist networks that capture propositional knowledge. *Artificial Intelligence*, 77:203–247.

Pratt-Hartmann, I. (2004). Fragments of language. *Journal of Logic, Language and Information*, 13(2):207–223.

Pudlak, P. (1999). A note on applicability of the incompleteness theorem to human mind. *Annals of Pure and Applied Logic*, 96(1-3):335–342.

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

Ristad, E. S. (1993). *The Language Complexity Game*. Artificial Intelligence. The MIT Press.

van Rooij, I. (2008). The tractable cognition thesis. *Cognitive Science: A Multidisciplinary Journal*, 32(6):939–984.

Sandler, U. and Tsitolovsky, L. (2008). *Neural Cell Behavior and Fuzzy Logic*. Springer, New York.

Shanahan, M. (1997). *Solving the Frame Problem: A Mathematical Investigation of the Common Sense Law of Inertia*. MIT Press, Cambridge, MA.

Siegelmann, H. T. (1995). Computation beyond the turing limit. *Science*, 268:545–548.

Siegelmann, H. T. (2003). Neural and super-turing computing. *Minds and Machines*, 13:103–114.

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

Simon, H. A. (1957). *Models of Man: Social and Rational*. John Wiley and Sons, Inc., New York, NY.

Simon, H. A. and Simon, P. A. (1962). Trial and error search in solving difficult problems: Evidence from the game of chess. *Behavioral Science*, 4(2):425–429.

Smolensky, P. (1987). Connectionism and cognitive architecture: a critical analysis. *Southern Journal of Philosophy*, 26 (supplement):137–63.

Smolensky, P. (1988). On the proper treatment of connectionism. *Behavioral and Brain Sciences*, 11:1–74.

Smolensky, P. (1991). Connectionism, constituency, and the language of thought. In Loewer, B. and Rey, G., editors, *Meaning in Mind: Fodor and His Critics*, pages 201–227. Blackwell.

Stenning, K. and van Lambalgen, M. (2008). *Human Reasoning and Cognitive Science*. MIT Press, Cambridge, MA.

Sternberg, R. J. (2002). *Cognitive Psychology*. Wadsworth Publishing.

Syropoulos, A. (2008). *Hypercomputation: Computing Beyond the Church-Turing Barrier*. Monographs in Computer Science. Springer, 1st edition.

Szymanik, J. (2007). A comment on a neuroimaging study of natural language quantifier comprehension. *Neuropsychologia*, 45:2158–2160.

Szymanik, J. (2009). *Quantifiers in TIME and SPACE. Computational Complexity of Generalized Quantifiers in Natural Language*. PhD thesis, Universiteit van Amsterdam.

Szymanik, J. (2010). Computational complexity of polyadic lifts of generalized quantifiers in natural language. *Linguistics and Philosophy*, 33(3):215–250.

Szymanik, J. and Zajenkowski, M. (2010a). Comprehension of simple quantifiers. Empirical evaluation of a computational model. *Cognitive Science*, 34(3):521–532.

Szymanik, J. and Zajenkowski, M. (2010b). Quantifiers and Working Memory. In Aloni, M. and Schulz, K., editors, *Amsterdam Colloquium 2009, Lecture Notes In Artificial Intelligence 6042*, pages 456–464. Springer.

Tsotsos, J. (1990). Analyzing vision at the complexity level. *Behavioral and Brain Sciences*, 13(3):423–469.

Turing, A. M. (1950). Computing machinery and intelligence. *Mind*, 59:433–460.

Tversky, A. and Kahneman, D. (1983). Extensional versus intuitive reasoning: The conjunction fallacy in probability judgment. *Psychological Review*, 90(4):293–315.

van Emde Boas, P. (1990). Machine models and simulations. In van Leeuwen, J., editor, *Handbook of Theoretical Computer Science*, pages 1–66. MIT Press, Cambridge, MA, USA.

van Gelder, T. (1990). Compositionality: A connectionist variation on a classical theme. *Cognitive Science*, 14:355–364.

van Gelder, T. (1991). Classical questions, radical answers: Connectionism and the structure of mental representations. In Horgan, T. and Tienson, J., editors, *Connectionism and the Philosophy of Mind*, pages 355–381. Kluwer Academic Publishers.

van Rooij, I., Stege, U., and Kadlec, H. (2005). Sources of complexity in subset choice. *Journal of Mathematical Psychology*, 49(2):160–187.

van Rooij, I. and Wareham, T. (2007). Parameterized Complexity in Cognitive Modeling: Foundations, Applications and Opportunities. *The Computer Journal*, 51(3):385–404.

Vogels, T. P. and Abbott, L. F. (2005). Signal propagation and logic gating in networks of integrate-and-fire neurons. *The Journal of Neuroscience*, 25(46):10786–10795.

Wason, P. C. (1968). Reasoning about a rule. *Quarterly Journal of Experimental Psychology*, 20:273–281.

Wason, P. C. and Shapiro, D. (1971). Natural and contrived experience in a reasoning problem. *Quarterly Journal of Experimental Psychology*, 23:63–71.