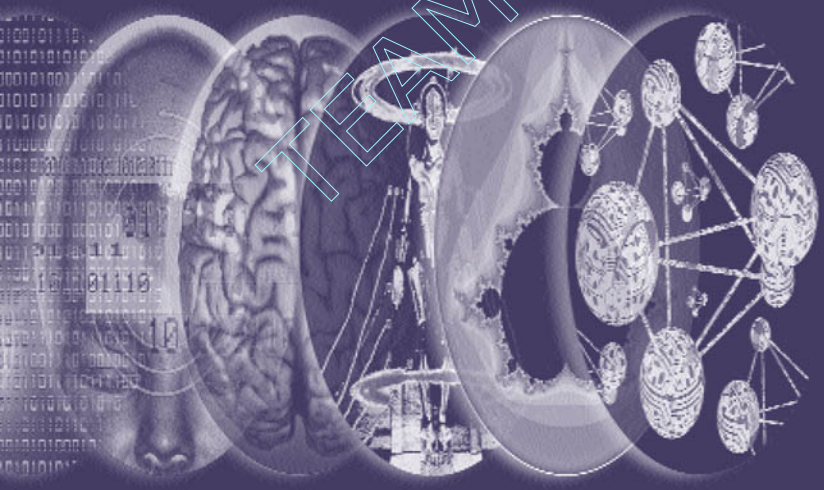
COMPUTATIONALISM

NEW DIRECTIONS

EDITED BY MATTHIAS SCHEUTZ



Computationalism

This page intentionally left blank

Computationalism

New Directions

edited by Matthias Scheutz

A Bradford Book
The MIT Press
Cambridge, Massachusetts
London, England

This book was set in Sabon by Achorn Graphic Services, Inc., on the Miles 33 system and was printed and bound in the United States of America.

Library of Congress Cataloging-in-Publication Data

Computationalism : new directions / edited by Matthias Scheutz.
p. cm.
"A Bradford book."
Includes bibliographical references and index.
ISBN 0-262-19478-3 (hc: alk. paper)
1. Computer science. 2. Artificial intelligence. I. Scheutz, Matthias.
QA76.C54747 2002
004—dc21 2002019570

Contents

This page intentionally left blank

Authors

Philip E. Agre, Department of Information Studies, University of California, Los Angeles, Los Angeles, CA 90095–1520, USA
pagre@ucla.edu
http://dlis.gseis.ucla.edu/pagre/

B. Jack Copeland, Philosophy Department, University of Canterbury, Christchurch, New Zealand
bjcopeland@canterbury.ac.nz
http://www.phil.canterbury.ac.nz/jack_copeland/

Stevan Harnad, Cognitive Sciences Center, ECS, Southampton University, Highfield, Southampton SO17 1BJ, United Kingdom
harnad@soton.ac.uk
http://cogsci.soton.ac.uk/harnad/

John Haugeland, Department of Philosophy, University of Chicago, Chicago, IL 60637, USA
j-haugeland@uchicago.edu

Matthias Scheutz, Department of Computer Science and Engineering, University of Notre Dame, Notre Dame, IN 46556, USA
mscheutz@cse.nd.edu
http://www.nd.edu/~mscheutz/

Aaron Sloman, School of Computer Science, The University of Birmingham, Birmingham B15 2TT, UK
axs@cs.bham.ac.uk
http://www.bham.ac.uk/~axs

Brian Cantwell Smith, Departments of Philosophy and Computer Science, Duke University, Durham, NC 27708–0402, USA
bcsmith@duke.edu
http://www.ageofsig.org/people/bcsmith/

This page intentionally left blank

Preface

Are minds computers? Or, to put it in more philosophical jargon, are mental states computational states? And if so, can human cognition then be understood in terms of programs? Computationalism—the view that mental states are computational states—is based on the conviction that there are program descriptions of mental processes and that, at least in principle, it is possible for computers, that is, machines of a particular kind, to possess mentality.

In its early days cognitive science rallied around computationalism, but in recent times this paradigmatic computational view of mind has come increasingly under attack. Connectionists and dynamicists have tried to replace it with alternative models. Biologists and neuroscientists have attempted to understand the mind directly at the level of the brain, thus skipping the "computational level." Social theorists and roboticists have argued that the essence of intelligence is to be found in situated interaction with the external world, rather than in a purely internal world of symbol manipulation. Philosophers have argued that traditional conceptions of computationalism (and more generally functionalism) are at best conceptually inadequate, if not vacuous (e.g., leading to the absurd view that any physical system can be viewed as implementing any computation).

Many of these critiques share a common theme. Computation fails as an explanatory notion for mind, the critics claim, because computation, assumed to be defined solely in abstract syntactic terms, necessarily neglects the real-time, embodied, real-world constraints with which cognitive systems intrinsically cope.

Although these views have led some researchers to abandon computationalism altogether, an increasing number is willing to reconsider

the very notion of computation, motivated in part by the recognition that real-world computers, like minds, must also deal with issues of embodiment, interaction, physical implementation, and semantics. This recognition raises the possibility that classical computationalism failed not because computing is irrelevant to mind, but because purely "logical" or "abstract" theories of computation fail to deal with issues that are vital to both real-world computers and minds. Perhaps the problem is not with computing per se, but with our present understanding of computing, in which case the situation can be repaired by developing a successor notion of computation that not only respects the classical (and critical) limiting results about algorithms, grammars, complexity bounds, and so on, but also does justice to real-world concerns of daily computational practice. Such a notion that takes computing to be not abstract, syntactic, disembodied, isolated, or nonintentional, but concrete, semantic, embodied, interactive, and intentional offers a much better chance of serving as a possible foundation for a realistic theory of mind.

Computationalism: New Directions is a first attempt to stake out the territory for computationalism based on a "successor" notion of computation. It covers a broad intellectual territory, from historic developments of the notions of computation and mechanism in the computationalist paradigm, to questions about the role of Turing machines and computational practice in artificial intelligence research; from different construals of computation and their role in the computational theory of mind, to the nature of intentionality and the origin of language.

The first chapter serves both as historic overview of the computationalist thinking and as introduction to the later chapters. It attempts to extract a historic trajectory that ties the mechanist views of past centuries to present perspectives on computation. Various references to later chapters point to places where the arguments are developed in more detail.

In the second chapter, Brian Smith examines various attempts to answer the question "what is computation?" Focusing on formal symbol manipulation and effective computability—two out of about a dozen different ways of construing "computation"—he shows that neither of them can do justice to the three conceptual criteria he sets forth. His investigation leads to the claim that "computation" is not subject matter and eventually to the demand for a new metaphysics.

B. Jack Copeland also points to a crucial distinction in chapter three, that between a narrow and a wide construal of "mechanism." The wide conception countenances the possibility of information-processing machines that cannot be mimicked by a universal Turing machine, allowing in particular the mind to be such a machine. Copeland shows that arguments for a narrow mechanism—the view that the mind is a machine equivalent to a Turing machine—are vitiated by various closely related fallacies, including the "equivalence fallacy" and the "simulation fallacy."

Chapter 4 takes on the issue of whether minds are computational in the Turing-machine sense from a quite different perspective. Here, Aaron Sloman criticizes the common view that the notion of a Turing machine is directly relevant to artificial intelligence. He shows that computers are the result of a convergence of two strands of historic developments of machines and discusses their relevance to artificial intelligence as well as their similarity to various aspects of the brain. Although these historic developments have nothing to do with Turing machines or the mathematical theory of computation, he claims they have everything to do with the task of understanding, modeling, or replicating human as well as animal intelligence.

In chapter 5 Phil Agre reveals five "dissociations," that is, intellectual tensions between two opposing conceptions such as "mind versus body," that have accompanied artificial intelligence (and computationalism) from its very beginning. He shows that although it is recognized that the two concepts underwriting each opposition are distinct, they are unintentionally conflated in the writings of the field. To overcome these difficulties, Agre advocates a "critical" technical practice that may be able to listen to and learn from reality by building systems and understanding the ways in which they do and do not work.

In chapter 6 Stevan Harnad, advocating a narrow conception of meaning, shows how per se meaningless symbols for categories are connected to what they mean: they are grounded in the capacity to sort, label, and interact with the proximal sensorimotor projections of their distal category-members in a way that coheres systematically with their semantic interpretations. He points out that not all categories need to be grounded this way and that language allows us to "steal" categories quickly and effortlessly through hearsay instead of having to earn them

through risky and time-consuming sensorimotor trial-and-error learning. It is through language that an agent (e.g., a robot) can acquire categories it could not have acquired through its sensors.

John Haugeland, then, broadens the discussion about meaning and intentionality in chapter 7 by providing a positive account of what is required for a "system" to have original intentionality, which he takes to be essential to genuine cognition. His main conclusion is that original intentionality presupposes an ability to accept responsibility. Thus, contrary to the assumptions of many researchers, responsibility is an essential topic of cognitive science, and the notions of intentionality and computation may both be explanatorily dependent on the notion of responsibility.

All seven chapters are completely self-contained and can, therefore, be read in any order. Common to all of them is the intention to initiate a discussion in an attempt to explicate, distill, and assess the foundations of cognitive science, rather than quickly and prematurely accept or dismiss computationalism as a viable theory of the mind for whatever reason. For each chapter (except the first) a preceding editor's note provides a brief overview of what to expect. The epilogue, finally, reflects in a more speculative way on what the next steps may be in the development of a successor notion of computation in an attempt to isolate promising directions and topics for future research.

It is my hope that *Computationalism: New Directions* will contribute to the development and study of a "successor notion" of computation and the range of its possible applications in the computationalist paradigm. Such a notion will have to take into account issues such as the program-process distinction, the notion of implementation and questions of physical realization, real-time constraints and real-world interactions, the use and limitations of models, relations between concrete and abstract, the proper interpretation of complexity results, the relation between computation and intentionality, notions of "wide content" and "wide mechanism," notions of locality and causation, virtual machines and architecture-based concepts, and many more. By addressing these and other questions so crucial to a firm foundation for cognitive science in this new century, this book is meant to be an invitation to philosophers and scientists alike to engage in and further this discussion.

Finally, I do not want to miss the opportunity to express my gratitude to the many without whom the book would not have become a reality.

I would especially like to mention and thank the authors of the various chapters for their contributions, the participants of the NTCS'99 conference in Vienna, "Computationalism: The Next Generation," for all the stimulating discussions, Leopold Stubenberg, Markus Peschl, my wife Colleen Ryan-Scheutz and many others for their critical comments on the various drafts of my contributions, Thomas Mayer for casting the topics of this book in colorful pixels for the book cover, and Robert Prior and Judy Feldmann from MIT Press for their editorial support. The book would not have been the book it is without them.

1

Computationalism—The Next Generation

Matthias Scheutz

The mind is to the brain as the program is to the hardware.
—P. N. Johnson-Laird

1 Generations: Computationalism and *Star Trek*

Everyone familiar with current science fiction will most likely recognize the origin of the attribute "the next generation" in the title of this chapter; it is borrowed from the *Star Trek* saga. So what, you might ask, do computationalism and *Star Trek* have in common? For one, both apparently have a "next generation," and furthermore, I would speculate, both "next generations" share the same fate regarding their initial popularity.

What I mean by this analogy is this: when *Star Trek: The Next Generation* came out, every *Star Trek* fan I know missed the old *Starship Enterprise,* its old crew, and in general the old technology, despite the fact that the new Enterprise was faster, the new crew smarter, and the new technology more advanced. It took some time to get used to the new frontiers and to appreciate the new show's potential. Once it grew to be appreciated, however, fans reflected on the limitations of the original series with a smile, perhaps even belittling the flashing lamps on the console indicating that the computer was performing some complex computation.

The next generation of computationalism might just be in a similar situation: most computationalists will probably not like it at first glance, for various reasons. Maybe because the new notion of computation it involves will be too broad in their view, or maybe because it will place emphasis on practical feasibility as opposed to theoretical possibility.

Whatever one's reason for a cautious confrontation with a successor version of computationalism may be, I would hope that the additional explanatory power slumbering in such an extension would trigger, as in *Star Trek,* the same "next generation" effect.

Nevertheless, there is a major difference between both "next generations" as far as their status quo is concerned: *Star Trek*'s successor has been produced already, whereas the new computationalism is still in the making. To get an idea of where this rebound of cognitive science's main view on mind—the "next generation of computationalism"—might be heading, I first trace the roots of computation to the seventeenth century to expose the strong, original bond between mind and computation. Then, after sketching the main tenets of the "old generation" of computationalism, I offer but a glance at some of the shortcomings for which computationalism has been criticized, interspersed with a list of issues that a successor notion of computation will have to address. Since this chapter is intended to be largely introductory, I have included references to subsequent chapters wherever appropriate—using only the last name of the respective author—for more detailed discussions of the various aspects of a "new computationalism."[1]

2 Mind as Mechanism

The notion of computation, although most prominently visible and increasingly entrenched in human culture these days, is not an invention of our times, despite the plethora of computing devices that have become part of our daily lives. Rather, it dates back to the seventeenth century and before, when different kinds of *mechanisms* were constructed to control the behavior of various types of machines (from looms, to organs, to watches and clocks). In particular, the first functioning *mechanical calculators* were built at that time: from Schickard's "calculating clock," to the first adding machine constructed by Pascal, to Leibniz's "Stepped Reckoner," and others, some of which are still in working order today (e.g., see Williams 1997 or Augarten 1985). Composed of mechanical parts like wheels, gears, springs, cogs, levers, etc., they employed the technology developed by watchmakers and were able to perform simple numerical operations such as the addition of two decimal numbers. Note

that these machines were far from being *autonomous* (in that they did not have their own energy source) or *automatic* (in that they could not perform operations by themselves). To use Sloman's distinction, both *energy* and *control information* had to be provided by humans for them to function properly.

While the potential of mechanisms for the construction of new kinds of machines was generally recognized, philosophers (like Descartes, Hobbes, Leibniz, La Mettrie, d'Holbach, and others) realized another potential of "mechanism": that of *mechanistic explanation.* As Copeland shows, parts of the human body were described in terms of analogies to mechanical parts (such as springs and wheels) and the behavior of the whole body explained in terms of mechanistic principles (e.g., Descartes). For some (e.g., La Mettrie) this explanation did not have to stop at the body, but could be further extended to the human mind (e.g., because the mind was viewed as organized in correspondence to parts of the body, i.e., the brain, which, in turn, could be explained in mechanistic terms). *In nuce,* the *mind* was viewed as a *machine,* giving rise to what Copeland calls "historical mechanism."

As Sloman points out, mechanical calculators are, in some sense, special kinds of mechanisms, since they are used not to control other machines or mechanical devices, but rather to perform calculations, that is, operations on numbers. However, since numbers are abstract entities, calculations cannot be performed directly on them, but have to be mediated through something physical that can be manipulated. Typically, these mediators were found in physical objects whose physical properties obey laws that are governed by operations that correspond systematically to the ones performed in the calculation. For example, the property "mass" is "additive": put two objects on a scale and their respective masses add up (this is why we are interested in an operation like "addition" in the first place). Hence, by correlating the magnitudes of a physical dimension (e.g., the length of physical objects) with numbers, calculations can be performed by physically manipulating objects (e.g., arranging them in a line) and then measuring the resulting magnitude (e.g., measuring the total length using a ruler). It was not until Vieta had introduced the concept of "representatives" that these slow and error-prone operations gave way to operations using *representations* (in

modern terminology, a transition was made from "analog" to "digital" representations). The advantages of *marks* to *stand in* for numbers (over correlating them with physical magnitudes of objects) led to a rapid expansion of this method for carrying out calculations with representations and eventually became a paradigm for thought in general (Pratt 1987). In the end, this rise of modern mathematics supported the view (due to Descartes, Hobbes, Leibniz, Locke, and others) that not only calculating, but thinking in general involves *the use and manipulation of representations*. In its most radical form, this idea can be and has been summarized by saying that "everything done by our mind is a computation" (Hobbes 1994, p. 30).

3 To Reason Is to Calculate . . .

In presenting these two views on mind that figure crucially in the origins of computationalism (the "mind" as some sort of a machine, e.g., like a calculator, and "thinking" as involving "the manipulation of representations"), I have not specified what kind of machine mind is taken to be, nor what kinds of manipulation thinking is supposed to involve. Although the first question is left unanswered by historical mechanists, the second may find an answer in Leibniz's conception of a "mechanical reasoner," a mechanical system that performs logical reasoning without any human aid.

Two important ideas underwrite the possibility of a mechanical reasoner: (i) that reasoning involves the manipulation of representations; and (ii) that logic can be viewed as a formal, deductive system in which reasoning takes the form of deductions that proceed according to rules. The first idea is intrinsically connected to Leibniz's view on concepts and language that there are simple representations from which all complex representations are built, but which themselves cannot be analyzed in terms of other representations—Harnad would call these simple representations "grounded." As an aside, one of the challenges in cognitive science nowadays is to give an account of these simple representations, what they are, where they come from, and how they can be used (see also Harnad's chapter).

The second idea is crucial to "mechanizing reasoning": it is by virtue of viewing logic as a formal, deductive system that valid principles of

reasoning can be formulated as rules of deduction, and once principles of reasoning are cast in the form of such rules, we can apply them directly to representations without having to know what the representation is *a representation of* and without having to justify the validity of the conclusion. Hence a mechanism constructed to take "representations" (of whatever form) and apply rules to them (similar to the calculating machines that perform operations on representations of numbers) would be able to reason by virtue of mere "calculations." Leibniz was even convinced that his formal method of reasoning would resolve any philosophical disagreement, for once a statement is formalized, its validity can be checked *mechanically:*

> There would be no more need for disputation between two philosophers than between two accountants. For it would suffice to take their pencils in their hands, to sit down to their slates, and say to each other (with a friend to witness, if they liked): calculemus—let us calculate. (Leibniz 1875–90, p. 200)

Note that Leibniz's mechanical reasoner is an early example of what Haugeland (1985) calls "automatic formal system," and that, furthermore, his view of calculating (e.g., as employed in the mechanical reasoner) already hints at the modern computationalist proposal that as long as the syntax is right, semantics will take care of itself (e.g., see Haugeland 1981a).

The above sketch of views of and relationships between mind, mechanism, and reasoning is—besides functioning as a brief historic exposition of the mechanist thinking—intended to introduce a suggestion I would like to make with respect to the origins of computationalism: the core ideas of present day computationalism can already be found in the seventeenth century. Even though their ideas were not expressed in today's terminology and are not based on our modern understanding of the notion of computation, the early "computationalists" (or historical mechanists) realized that the notion of computation effected a link between the mental and the physical. This is because the notion of computation was intrinsically connected to the operations performed by mechanical calculators, on one hand, and to cognitive processes using representations (such as calculating and reasoning), on the other. It was this link that eventually gave rise to the hypothesis that *mind* might be *mechanizable.*

4 The Separation of Mind and Mechanism

It is important to be aware of the relationships between mind, computation, and mechanism, which were viewed as intrinsically intertwined by the early "computationalists," to be able to appreciate the developments of computationalism in centuries to follow. For one thing, the materialism implied by the early computationalists was overshadowed by various forms of German idealism, which took the mental and not the physical to be primary and whose adherents were by no means open to mechanistic explanations. Not surprisingly, the mechanistic view of mind was not very popular in the eighteenth and nineteenth centuries except for a few convinced materialists (e.g., the German physiologists Vogt, Büchner, or Moleschott).

The last century, however, witnessed a major advance in both the construction of computing devices and the conception of computation. While many attempts were made at building mechanical calculators up to the end of the nineteenth century (some of which were quite successful; e.g., see Pratt 1987, Williams 1997, or Augarten 1985), the computing capabilities of these mechanical devices remained extremely modest compared to the electronic computers we know today, a development initiated by the rapid progression in the engineering of electronic components (from vacuum tubes, to transistors, to integrated circuits, and beyond; e.g., see Williams 1997).[2] Similarly, the notion of computation (as used by the early computationalists) remained at an intuitive level until it became the focus of logical analysis, stimulated by investigations of the notions of "formal system" and of "demonstrability" (i.e., proof by finite means) of formulas in formal systems, which, in turn, led to further studies of notions such as "recursive function," "effectively computable function," "algorithm," "finite state automaton," and others in the first half of the twentieth century.

It was at about this time that the notion of computation "split" and took off in two directions, each of which led to a particular view on and interest in computation. The "logical" or "theoretical" route was concerned with a logical explication of the intuitive notion of computation and theoretical limitation results of what could be computed in principle, whereas the "*techno*-logical" or "practical" route was very much

focused on all sorts of problems connected to producing reliable computing devices that could be used in a variety of contexts, from scientific computing to military and commercial applications, and that would satisfy the increasing demand for fast and powerful machines.

Although both approaches are concerned with important aspects of computation and are by no means incompatible, their use of the term "computation" is subject- and interest-specific. Moreover, since research can be conducted quite independently in both disciplines, it is not surprising that these two routes did not cross very often in the past. Only in more recent times do we witness a mutual interest, as logic became more sensitive to real-world constraints (complexity theory, for example). Alternative conceptions of computations such as interactive Turing machines, games, and so on are thought to overcome the separation and dissociation of classical logical models from worldly concerns.

5 Cognitive Science or the Rebirth of Computationalism

This separation of computation into two quite independent notions somewhat parallels the separation of mind and mechanism that had taken place earlier at various times in the history of philosophy, but most notably with Descartes. While it enabled people to talk about computations without the need to refer to the particular mechanism that carried them out, it introduced an explanatory gap between computations *qua computations* and what does the computing, i.e., the *mechanism* or *computer,* eventually leading to what Smith calls the *mind-body problem for machines:* how are computations related to computers? (I will come back to this problem in section 9.)

The independence of computations from their physical realizers, however, was one major source of attraction for some psychologists in the late 1950s. Another was the potential of computers to process information—an ability thought to be crucial to human cognition. Together they gave rise to a powerful metaphor, often called the "computer metaphor," that "the mind is to the brain as the program is to the hardware" (Searle 1980; Johnson-Laird 1988).[3] It is this computer metaphor that underwrites the rebirth of computationalism in the twentieth century, and the birth of what is nowadays known as *cognitive science* (e.g., see Gardner

1985): by viewing cognitive functions as computations, explanations of mental processes in terms of programs become scientifically justifiable without having to take neurological underpinnings into account—the "wetware brain" is simply viewed as a computer on which the software "mind" is running (or if not mind itself, then at least all the cognitive functions that constitute it).

Pinpointing the various positions and claims subsumed under the notion of computationalism would be a research project in its own right. Just to give you an idea of what can be found in the literature, sloganlike phrases such as "the brain is a computer," "the mind is the program of the brain," "cognition is computation," or "the mind is a computer" are not uncommon, and these are only a few. Note that in a phrase like "cognition is computation" the interpretation of every single word matters, "is" included—do we interpret "is" as "extensional identity" or "extensional inclusion"? Or do we read it intensionally? Such statements are necessarily condensed and cannot be taken at face value; for if they were read together, essentially distinct notions (such as program and process, mind and cognition) would be equivocated.

There are other descriptions of computationalism that emphasize the information-processing capabilities of computers. For example, computationalism has been characterized as the conjunction of the theses "thinking is information processing," "information processing is computing (i.e., is symbol manipulation)," and "the semantics of those symbols connect mind and world." Again others emphasize the reliance on logical notions of computations. Dietrich (1990), for example, takes computationalism to be "the hypothesis that cognition is the computation of functions," which "makes no claims about which functions are computed, except to say that they are all Turing-computable (computationalists accept the Church-Turing Thesis), nor does it make any specific claims as to how they got computed, except to say that the functions are systematic, productive, and interpretable in a certain way" (p. 135). Since many computationalists seem to be (or have been) content with notions of computation as provided by formal logic, it is helpful to know the interest of logicians in computation and to understand the motivations and results of the "logical route" to be able to put the logical contributions to and possible influence on twentieth-century computationalism into perspective.

6 Logic and Computation: The Rise of Turing Machines

The logical side of the history of computation started in the 1930s with various attempts to make the intuitive notion of computation (then called "effective calculability") formally precise. It was solely concerned with what could be computed *in principle,* which, in turn, required a thorough analysis of the *intuitive notion of computation.* The most crucial insight of the 1930s with respect to the meaning of this intuitive notion of computation was most likely the fact that three different attempts to characterize it formally could be proven equivalent: the class of recursive functions equals the class of λ-definable functions equals the class of Turing-machine-computable functions. These equivalence results are possible because what "computing" means with respect to any of the suggested formalisms is expressed in terms of functions from inputs to outputs; and using functions as mediators, the different computational formalisms can be compared according to the class of functions they compute.

Later, other formalisms such as Markov algorithms, Post production systems, universal grammars, PASCAL programs, as well as various kinds of automata were also shown to "compute" the same class of functions, referred to as *recursive functions* (e.g., see Hopcroft and Ullman 1979). The extensional identity of all these formalisms supports a definition formulated by Church, which later became known as "Church's Thesis":

> We now define the notion [. . .] of an effectively calculable function of positive integers by identifying it with the notion of a recursive function on positive integers (or of a λ-definable function of positive integers). This definition is thought to be justified by the considerations which follow, so far as positive justification can ever be obtained for the selection of a formal definition to correspond to an intuitive notion. (Church 1936b, p. 356; also in Davis 1965, p. 100)

Using the various equivalence results it follows from "Church's Thesis" that any of the above-mentioned formalisms captures our intuitive notion of computation, that is, *what it means to compute.* Although this thesis cannot be proved in principle as mentioned by Church himself, it became more and more plausible as newly conceived computational formalisms were shown to give rise to the same class of "computable" functions.

What is common to all these computational formalisms, besides their attempts to specify formally our intuitive notion of "computation," is their property of being independent of the physical. In other words, computations in any of these formalisms are defined *without* recourse to the nature of the physical systems that (potentially) realize them. Even Turing's machine model, the so-called Turing machine, which is considered the prototype of a "mechanical device," does not incorporate physical descriptions of its inner workings, but abstracts over the mechanical details of a physical realization.

Interestingly enough, Turing (1936) invented his machine model of "computation" in order to capture the human activity of "computing," that is, the processes a person (the "computer") goes through while performing a calculation or computation using paper and pencil. Although he used the term "computer" to refer to whatever is doing the computations, he did not intend it for a digital computer, but for a human person doing computations—at that time digital computers did not yet exist.

In Turing's analysis of the limitations of the human sensory and mental apparatus, five major constraints for "blindly" following rules to do computations crystallize (I follow the presentation in Gandy 1988):[4]

1. Only a finite number of symbols can be written down and used in any computation.
2. There is a fixed bound on the amount of scratch paper (and the symbols on it) that a human can "take in" at a time in order to decide what to do next.[5]
3. At any time a symbol can be written down or erased (in a certain area on the scratch paper called "cell").
4. There is an upper limit to the distance between cells that can be considered in two consecutive computational steps.
5. There is an upper bound to the number of "states of mind" a human can be in and the current state of mind together with the last symbol written or erased determine what to do next.

Although there are certainly some steps in Turing's analysis of an abstract human being performing calculations that seem rather quick and not well supported, one can summarize the above in Gandy's words as follows: "The computation proceeds by discrete steps and produces a record consisting of a finite (but unbounded) number of cells, each of which is blank

or contains a symbol from a finite alphabet. At each step the action is local and is locally determined, according to a finite table of instructions" (1988, p. 81).

In other words, by "abstracting away" from persons, scratch paper, and so on, Turing claimed that all "computational steps" a human could possibly perform (only by following rules and making notes) could also be performed by his machine. In this way the Turing machine became a model of human computing, an *idealized* model to be precise, since it could process and store arbitrarily long, finite strings of characters. It is worth pointing out that Turing, as opposed to Church, did not only state a "thesis" regarding the intuitive notion of computation, but actually intended it as a theorem (see also Gandy 1988, p. 83, who restates Church's Thesis as Turing's Theorem): "Any function that can be computed by a human being following fixed rules, can be computed by a Turing machine."

Turing also believed the converse, that every function computed by a Turing machine could also be computed by a human computer, although this, again, does not take time and space restrictions seriously, but rather assumes an abstract human computer not subject to such worldly limitations. In particular, Turing was convinced that "the discrete-state-machine model is the relevant description of one aspect of the material world—namely the operation of brains" (Hodges 1988, p. 9; see also Copeland's chapter). The origins of Turing's claim can be found in the intrinsic connection between the notion of "computability" and Gödel's notion of "demonstrability" (of a proof in a formal system): that which can be "demonstrated" using "definite methods" amounts to what can be done by a Turing machine (see Turing 1936). By relating the limitations of formal systems as pointed out by Gödel to the limitations of his machine model, Turing "perceived a link between what to anyone else would have appeared the quite unrelated questions of the foundations of mathematics, and the physical description of mind. The link was a scientific, rather than philosophical view; what he arrived at was a new materialism, a new level of description based on the idea of discrete states, and an argument that this level (rather than that of atoms and electrons, or indeed that of the physiology of brain tissue) was the correct one in which to couch the description of mental phenomena" (Hodges 1988, p. 6).

7 Strong AI—The Heyday of Computationalism

Ever since its conception, the notion of a Turing machine has been used in the metatheoretical discourse of AI and cognitive science for two opposing purposes: either to justify the use of computers to model, simulate, or duplicate cognition as underwriting computationalism, or to argue that minds are not Turing machines (e.g., see King 1996). This philosophical tension is reflected in literature throughout AI's (and cognitive science's) fifty-year history. Copeland, for example, shows how various deviants of the Church-Turing Thesis as well as certain statements by Turing himself have been mistaken as claims about the nature of mind. In accordance with Agre, one could add this tension—is the human mind a Turing machine?—to the many other foundational discrepancies, which have been tacitly imported into AI from the surrounding intellectual territory without having ever been digested.

Despite and mostly independent of this and other unresolved philosophical debates (see Agre's and Sloman's chapters), AI based on the *computer metaphor* (also called *strong AI* or *GOFAI,* for "good old-fashioned AI"), has been quite a successful project, which already early on produced many impressive programs (from Newell, Shaw, and Simon's *Logic Theorist* and *General Problem Solver,* to Samuel's checkers program, to Bobrow's *Student,* to Weizenbaum's *Eliza* and Colby's *Parry,* to Winograd's *SHRDLU,* to Lenat's *Automated Mathematician,* to various expert systems like *DENDRAL, MYCIN,* and *XCON* and other game playing programs; e.g., see Crevier 1993).

Two main assumptions are buried in this metaphor: (1) that the mind can somehow be "understood as a computation" or be "described by a program" (this requires the adoption of a notion of computation or program, respectively); and (2) that the same kind of relation that obtains between computational processes (i.e., executed programs) and computer hardware—the implementation relation—obtains between minds and brains, too. While assumption (1) led to fertile research in artificial intelligence and cognitive science, which in turn has been taken as evidence for its truth, assumption (2) by and large remained at the level of an assumption.[6]

Expanding a bit on the first assumption to get a better idea of how to think of the mind as being described by a program, note that computation

can be viewed as the *rule-governed manipulation of representations,* very much in line with the views of the early computationalists. After all, this is what computers do: they manipulate symbol tokens (e.g., data structures implemented in sequences of bits), which themselves are representations of the subject matter the computation is about (compare this to Newell's 1980 notion of "physical symbol system"). These representations, in turn, have both *formal* and *semantic* properties, of which only the former are *causally efficacious,* that is, can do "real work" in the system. This view of computational processes as manipulating symbols by virtue of their formal and not their semantic properties—call it "formal symbol manipulation"—is predominantly found in the philosophical literature (e.g., Fodor 1989 or Haugeland 1985), but summarizes nicely what is appealing (or appalling, depending on one's point of view) about computationalism: that the *semantics* of symbols, or more generally, *intentionality* plays no role in cognitive processing, leading to the hope that computation might be the cornerstone for a theory of consciousness (e.g., see Dennett 1991) and intentionality (see Smith's and Haugeland's chapters).

Computationalism has many appealing facets, especially when it comes to high-level cognition: many features related to logic and language (such as systematicity, productivity, compositionality and interpretability of syntax or the compositionality of meaning; e.g., see Fodor and Pylyshyn 1988) are supported by computations "almost for free," and many mental operations on various kinds of representations such as rotating three-dimensional images, searching for a good move in a chess game, reasoning about other people's behavior, or planning a route through a city avoiding construction sites can be described computationally and implemented on computers.

In sum, the crucial factors making the notion of computation an attractive candidate for explaining cognition include the potential of computations to have semantics, while being causally efficacious, algorithmically specifiable (in programming languages), and implementable in digital computers.[7]

8 Computationalism under Attack

Computationalism has always been under attack from various directions even before it was officially recognized as such (if not for anything else,

then for the mere fact that for some it was inconceivable that mind could be mechanized; especially, for any dualist, who believes that the *mental substance* is different from the physical, the computationalist paradigm is unacceptable). Yet, the possibility of using formal methods to characterize the notion of computation, especially the notion of "effective procedure," enabled critics of computationalist views to utilize results from formal logic in order to make their points more rigorously. At the beginning of the 1960s, for example, Lucas (1961) used Gödel's incompleteness theorems to argue that the mind cannot be a Turing machine. The ensuing intellectual debate about what is right and what is wrong with this claim had various repercussions throughout the subsequent decades, most recently spurred by Penrose (1989, 1994) (e.g., see the various comments on his books in *Behavioral and Brain Sciences* 13 and *PSYCHE* 2).

Various other objections, not based on formal logic, such as problems with the notion of representation, wide notions of meaning, content and supervenience, universal realization results, and so on have been advanced by philosophers over the years and debated at great length (e.g., see Putnam 1975, 1988; and Dreyfus 1979, 1992). One particularly famous and still widely discussed problem is Searle's Chinese room thought experiment (e.g., see Searle 1980 and its various commentaries; Searle 1990; and also Harnad 2001b).

More recently, connectionists have tried to broaden (if not replace) the notion of computation with alternatives arguing that the symbolic/computational level of description so crucial to computationalism cannot be taken for granted. Others—biologists and neuroscientists, for example—are trying to "go in under" the computational level to understand the mind directly at the level of the brain. Diversely, some social theorists and roboticists have argued that the essence of intelligence is to be found in situated interaction with the external world, rather than in a purely internal world of symbol manipulation. Harnad, for example, calls such "disconnected," internal symbols "ungrounded" and shows how it is possible in a robotic system to ground some of its internal symbols (in sensorimotor projections of distal objects), by virtue of which other "ungrounded" symbols can obtain their "meaning."

While some connectionists believe that symbolic activity should emerge from a "subsymbolic level" (e.g., Smolensky 1988), most of the so-called dynamicists (e.g., Port and van Gelder 1995) find the symbolic level of description superfluous altogether and argue instead for an explanation of cognition in terms of dynamic systems.[8] For example, van Gelder (1998) argues that the notion of an effective procedure is essential to computation, and that essential to the notion of an effective procedure, in turn, is the notion of discrete steps in an algorithm. He claims that this discreteness, in both its temporal and nontemporal aspects, prevents computation from explaining many aspects of cognition, which he considers to be a fundamentally dynamical phenomenon. Hence, instead of using any of the standard computational formalisms, one ought to use *dynamical systems* in describing cognitive functions, the idea being that every real-world system (and thus cognitive systems as well!) involving change can potentially be modeled by a dynamical system—this is what dynamic systems have been designed to do. According to the respective system, this will happen at different levels of description, at the very low level of fields (consider Maxwell's equations), or the very high level of human decision making (e.g., consider the decision field theory by Busemeyer and Townsend 1993).[9]

Another attack, also advanced by dynamicists, challenges the role of representation in cognitive science in general, and a fortiori can be seen as a challenge to the role of computation in cognitive science. Especially psychologists have argued that certain allegedly "cognitive" tasks have nothing to do with cognition proper, but are really motor control tasks that can be explained and modeled in the language of dynamical systems without resorting to manipulations of representations (e.g., see Smith and Thelen 1994). As a consequence, the following question arises: to what extent do notions of representation have to be involved in explaining cognitive abilities, and furthermore, is it possible to invoke "representation" within dynamical system theory itself when needed, thus bypassing the "classical representational=computational" level of description (see the various articles in Port and van Gelder 1995)?

Finally, there are criticisms concerning the very founding notions of computationalism: the notion of computation and its conceptual complement, the notion of implementation, which are presented next.

9 The Recalcitrant Notions of Computation and Implementation

As Smith argues, traditional notions of computation underwriting computationalism are conceptually inadequate and at best incomplete, revealing crucial hidden assumptions and conceptual dependencies, on which various construals of "computation"—from "formal symbol manipulation," to "effective computability," to "execution of an algorithm"—are based (see also Smith 1996, and forthcoming). One main consequence is that computation cannot provide a theory of intentionality (as hoped by computationalists), but rather seems to depend on it. Haugeland, for example, exposes deep and complex relationships and dependencies between intentionality and responsibility, which have been largely ignored by computationalists, and which need to be accounted for in computational terms if computationalism is to succeed as an explanatory theory of mind.

To get a feeling for the kinds of problems that arise from classical notions of computation, take the view of computation as "computation of an input-output function" (e.g., as proposed by Dietrich 1990) and try to answer the following questions: can every computation be explained as the computation of a function? Consider, for example, arcade video games. What input-output functions do they compute? Or take operating systems. They are specified by programs, which are designed to never halt. Contrast this then with the classical approach, where such so-called divergent functions (i.e., functions that are not defined for certain arguments) are neglected. Further questions about *how* a function is computed arise: does computing a function imply "following rules" or "executing an algorithm"? Does the computation of a function have to be *productive,* or would nonalgorithmic methods that arrive at the same results count as computing the function too (a big "look-up table," for instance)? And, in general, how can we say that a particular physical system computes a given function, that is, that the system implements a particular computation?

Most computationalists tacitly assume that the notion of implementation (as used in computational practice) is unproblematic; it is quite common to think of implementation as some sort of correspondence between computational and physical states. For example, there is a tight correspondence between parts of the architecture of a von Neumann CPU and

expressions of the assembly language for that CPU. Another example of how computational descriptions can "mirror" physical descriptions is a logic gate (e.g., an AND gate), whose "computational capacity" is described by a *Boolean function,* the values of which in turn can be related to physical magnitudes in the physically realized circuit.

Although we may be able to establish a functional correspondence between physical and computational states for certain artifacts (i.e., devices we have designed to allow for computational descriptions), it is unclear that this can be done for natural kinds (such as brains) too. In particular, one has to be aware that any such correspondence crucially depends on "adequate" physical states:[10] computations "mirror" the causal structure of a system under a given correspondence function between computational and physical states only relative to the choice of the physical states. Computational explanations of the behavior of a given physical system, therefore, depend essentially on those physical states of the system that can be set in correspondence to some computation. This dependence, per se, is not problematic as long as one can assume the system has "appropriate physical states" (e.g., as in the case of electronic devices). If, however, it could be shown that for any computational description and any given physical system, one can find "physical states" of that system that can be set in correspondence with the computational ones and are, furthermore, appropriate (in a certain sense of "appropriate" that depends on the underlying physical theory), then computational explanations would be in danger: every system could then be said to compute. In other words, computationalism would be vacuous if every physical system could be viewed as implementing every computation.

And, indeed, it has been argued that, assuming an intuitive notion of implementation, any (open) physical system can be seen to implement any finite state automaton (Putnam 1988), or that, for example, walls implement the WordStar program (Searle 1992). If that is so, then something must have clearly gone wrong with our intuitions, since such a view of computation and implementation is not tenable, either from the theoretician's or from the practitioner's perspectives.

Many people have attacked these claims by finding flaws in the arguments; some have even attempted positive accounts of what it means for a physical system to implement a computation (e.g., Chalmers 1996; Melnyk 1996). However, even these revised accounts have problems of their

own (e.g., see Scheutz 2001), which in the end might require an account of implementation that is based not on the idea of "mirroring" computations in the physical system, but rather on a series of abstractions over the physical peculiarities of the implementing system (Scheutz 1999).

10 A Rebound of Computationalism?

While all of the above-mentioned critiques of computationalism vary, they share a common theme: computation fails as an *explanatory notion* for mind, because computation necessarily neglects the real-time, embodied, real-world constraints with which cognitive systems intrinsically have to cope. This is a consequence of assuming computation to be defined solely in abstract syntactic terms (abstracting over physical realization, real-world interaction, and semantics).

How can we continue to hold on to computationalism, you may ask, when it seems that digitality is restrictive, formal symbol manipulation is not sufficiently world-involving, and Turing machines are universally realizable to the point of vacuity? It should not come as a surprise that these issues, combined with recent progress made by dynamicists (while the classical approach prima facie appears stagnant), have led many cognitive scientists to abandon computationalism altogether.

Despite the dire prospects for computationalism currently envisioned by some (especially convinced dynamicists), an increasing number of researchers are not willing to give up what has been *in nuce* a very promising and up to now partly quite successful approach. Inspired by their recognition of the fact that real-world computers, like minds, also deal with issues of embodiment, situated interaction, physical implementation, and semantics, they are motivated to reconsider the very notion of computation as well as the whole enterprise of computing and computational modeling. This motivation is at least partly based on the possibility of classical computationalism failing *not because computing is irrelevant to mind,* but because the classical, purely "logical" or "abstract" theories of computation, which were taken to be the fundaments of the "old generation computationalism," do not to address real-world aspects that are vital to both (real-world) computers and minds.

Artificial intelligence has never been out of touch with the practical, technological route of computation (as Sloman reminds us). Many of the

features of present-day computers employed in artificial intelligence research were originally inspired by features of (natural) brains, and conversely inspired theories about brains. Yet, this fact is usually ignored by any theoretical discourse that views computation as Turing-machine computability.

Furthermore, by (mis)interpreting and applying results from the logical route of computation, in particular the Church-Turing thesis, to argue that cognition is or is not Turing computation, Turing-machine computability is often equated with computability by a mechanism. However, as Copeland points out, there are *conceptual computing machines* that can outperform any Turing machine (e.g., Turing's O-machine—whether such a machine could possibly exist is a separate issue), which are compatible with the mechanistic views of the early computationalists (the operations of such machines can also be described algorithmically; e.g., see Shagrir 1997). Such computing devices undermine the common prejudice that computation is limited to "effective computation" (or Turing-machine computability).

Another part of the problems of computationalism may be the conceptual separation of mind and mechanism, as reflected in the separation of computation and what does the computing, a distinction that underwrites computationalism. It almost seems as if this split attempts to "hold something apart from and above material reality," to use Agre's words, who locates a general pattern of such "dissociations" permeating AI.

All of this seems to point in the same direction: that the current problems of computationalism do not so much lie in computing per se, but *in our present understanding of computing.* The notion of Turing computability (and with it other classical notions) may be part of the foundational problems of computationalism, if—as Smith suggests—the theory of Turing machines is not, contrary to current orthodoxy, a theory of computation in the first place, but rather a *theory of marks,* and hence *not* a theory of *intentional phenomena.* In fact, as Sloman's reconstruction of the historic development of computers and mechanical artifacts (especially within artificial intelligence research) shows, it may be irrelevant and not required for computationalism at all.

However, what is required and will have to be addressed by the next generation is the intrinsic relationship between intentionality and responsibility, which Haugeland makes clear in his positive account of

intentionality: for a system to be able to have *genuine intentionality* is for it to have the capacity to accept what he calls "authentic responsibility." A milestone along the way to implementing genuine intentionality may be to start with Harnad's suggestion of "grounding" symbols in sensorimotor capacities. Such grounded symbols can be used to form new complex symbol structures, which inherit their meaning systematically from their constituents, and may lie at the heart of language, and consequently, meaning.

At this point, I have explicated the status quo of the "old generation" and identified some of the goals for the "next generation" of computationalism. The contours of this "next generation" are already visible, but need to be fleshed out in detail. Now is the time for the chapters to take over, elaborating many of the above as well as other issues in a first attempt to rehabilitate what may still be our best bet for explaining cognition.

Notes

1. Some of the following material is based on "The Cognitive-Computational Story" in Scheutz (2000) with the permission of Academia Verlag, St. Augustin.

2. The timeline available at IEEE's web site demonstrates graphically the enormous gain in momentum of computer development ever since the first transistor was constructed in 1947, which led Gordon Moore in 1965 to make the prediction—now called "Moore's Law"—that the number of transistors per integrated circuit would double every eighteen months. Interestingly, Moore's Law is still true today.

3. Note that the computer metaphor should read "the mind is to the brain as *(computational) processes* are to the hardware" to avoid blurring the "program-process" distinction.

4. As Gandy puts it: "Turing's account of the limitations of our sensory and mental apparatus is concerned with perceptions and thoughts, not with neural mechanisms, and there is no suggestion that our brains act like Turing machines" (1988, p. 87).

5. This requirement does not exclude an arbitrary amount of scratch paper. It just delimits the range of perception, i.e., the amount of information the human "computer" can use at any given time to determine the next step in the computation.

6. This is so, presumably, because neither AI researchers nor psychologists need to pay attention to it. AI researchers, who build computational models, implicitly deal with the implementation relation of software on computer hardware on a daily basis, but are not concerned with the implementation relation of minds on

"brain hardware"; nor are psychological studies, which remain at the level of "program description." Neuroscience would probably be the closest discipline concerned with implementation issues of brains. Yet, neuroscientists do not attempt to relate programlike descriptions to brain areas; rather, they attempt to study the functional role of these areas (with respect to the rest of the brain) directly by virtue of their physiological functions. However, so-called computational neuroscientists already make use of "alternative models of computation," e.g., connectionist networks.

7. Compare this to Bridgeman's (1980) statement about the subject matter of artificial intelligence: "Artificial intelligence is about programs rather than machines only because the process of organizing information and inputs and outputs into an information system has been largely solved by digital computers. Therefore, the program is the only step in the process left to worry about."

8. Surprisingly, a common misperception among dynamicists seems to be that computationalism and "dynamicism" are mutually exclusive.

9. To describe a physical system, one needs to introduce a variable for each relevant physical dimension and consider it as a function of time. The simplest way to specify the behavior of the physical system would be to provide graphs of each such variable over time, that is, to have a set of functions $X_1(t)$, $X_2(t)$, . . . , $X_n(t)$ where $X_i(t)$ yields the "state" of the relevant physical dimension X_i at time t. This set of functions will determine the behavior of the system for all times. However, it does not reveal the possible dependencies of the X_i on each other. This is where differential equations come in handy. They provide a way of specifying the various interdependencies of variables in such a way that graphs of each variable can be obtained from them, yet the interdependencies are also brought to the open. The nature of these interdependencies becomes a crucial factor in an explanation of the behavior of the system, and the mathematical theory of dynamical systems seems well suited quantitatively to describe systems that exhibit such interdependencies.

10. In the case of electronic devices, the appropriate physical states can be determined either by asking the engineers who designed the devices or by looking at the blueprint and comparing it to the computational description of the device. In the case of biological systems, however, such states are not clearly defined. Consider, for example, physical states of a pyramidal cell: which of those states could correspond to computational states such that the respective computation captures essential parts of the causal structure of these cells? It has been suggested that "firing" vs. "not firing" would be "natural" candidates (e.g., by McCulloch and Pitts 1943), but it turns out that this computational model is too reductive, as it completely neglects essential temporal processes (such as temporal integration of signals, maximal firing rates, etc.). Hence more factors about pyramidal cells need to be taken into account, yielding more physical states that have to correspond to computational ones, etc. Artificial neural networks seem to be promising candidates for such computational descriptions, but to my knowledge the issue has not been resolved. It might well be that in the end the complex behavior of pyramidal cells defies a computational description, but this is obviously an empirical issue.

This page intentionally left blank

2

The Foundations of Computing

Brian Cantwell Smith

Editor's Note

What is computation? Not what current theories of computation say it is, argues Smith, as they one way or another "implicitly rely, without explanation, on such substantial, recalcitrant notions as representation and semantics," possibly even suggesting computation as a candidate for a theory of those very notions. Smith distinguishes various accounts of computation, originating in different intellectual areas and aiming at different goals. For example, there is the construal of computation as "formal symbol manipulation," embracing the idea of a machine manipulating symbolic or (at least potentially) meaningful expressions without regard to their semantic content. Or there is computation seen as the "execution of an algorithm," or the mathematical notion of "effective computability." Additional notions of computation include "digital state machine," "information processing," and "physical symbol system." All of these construals fail to meet at least one of three criteria, which a comprehensive theory has to satisfy, according to Smith. The first, an "empirical" criterion, requires theories of computation to do justice to real life computing, that is, to account for and be able to explain programs like Microsoft Word, what it does, how it is used, and so on. The second is a conceptual criterion, which requires a theory of computation to "discharge all intellectual debts" such as clarifying the relation between computation and various other notions it depends on or is related to. Finally, the third criterion concerns computation's role in computationalism in that it requires a theory of computation also to be an intelligible foundation for the formulation of the computational theory of mind (whether the latter is true or false is not at stake here). Computation, Smith suggests, is intrinsically intentional—this was what made computation an attractive aspect of computationalism in the first place. Yet, it is this intentional or semantic character of computation that is disguised by the widely held, pretheoretic conception of computation as being entirely formal. Once the involved notion of formality is scrutinized, however, it becomes clear that computation cannot be correctly classified by any reading of "formal," and hence the semantic character of computation is in need of explanation. So, rather than providing one, computation will have to wait for the development of a satisfactory theory of intentionality. But Smith does not stop here. Instead he calls into question the whole set of ontological assumptions underlying computational

analyses, culminating in his claim that computation is not a subject matter. Hence, although a satisfactory analysis of computation will have to include a theory of semantics and a theory of ontology, we will never have a "theory of computing," because computation does not constitute a distinct ontological or intellectual category. To some this may seem a negative conclusion, but for Smith it opens up the possibility of seeing computers as embedded in a rich practice, which might enable us to see "how intentional capacities can arise in a mere physical mechanism."

1 Introduction

Will computers ever be conscious? Is it appropriate—illuminating, correct, ethical—to understand people in computational terms? Will quantum, DNA, or nanocomputers require radical adjustments to our theories of computation? How will computing affect science, the arts, intellectual history?

For most of my life I have been unable to answer these questions, because I have not known what computation is. More than thirty years ago, this uncertainty led me to undertake a long-term investigation of the foundations of computer science. That study is now largely complete. My aim in this chapter is to summarize a few of its major results.[1]

2 Project

The overall goal has been to develop a comprehensive theory of computing. Since the outset, I have assumed that such an account must meet three criteria:

1. *Empirical:* It must do justice to—by explaining or at least supplying the wherewithal with which to explain—the full range of computational practice;
2. *Conceptual:* It must as far as possible discharge, and at a minimum own up to, its intellectual debts (e.g., to semantics), so that we can understand what it says, where it comes from, and what it "costs"; and
3. *Cognitive:* It must provide an intelligible foundation for the computational theory of mind: the thesis, often known as *computationalism,*[2] that underlies traditional artificial intelligence and cognitive science.

The first "empirical" requirement, of doing justice to practice, helps to keep the analysis grounded in real-world examples. By being comprehensive in scope, it stands guard against the tendency of narrowly defined

candidates to claim dominion over the whole subject matter.[3] And it is humbling, since the computer revolution so reliably adapts, expands, dodges expectations, and in general outstrips our theoretical grasp. But the criterion's primary advantage is to provide a vantage point from which to question the legitimacy of all extant theoretical perspectives. For I take it as a tenet that what Silicon Valley *treats* as computational *is* computational; to deny that would be considered sufficient grounds for rejection. But no such a priori commitment is given to any *story* about computation—including the widely held recursion- or Turing-theoretic conception of computability, taught in computer science departments around the world, that currently lays claim to the title "The Theory of Computation."[4] I also reject all proposals that assume that computation can be *defined*. By my lights, that is, computer science should be viewed as an empirical endeavor.[5] An adequate theory must make a substantive empirical claim about what I call *computation in the wild*:[6] that eruptive body of practices, techniques, networks, machines, and behavior that has so palpably revolutionized late twentieth- and early twenty-first-century life.

The second, "conceptual" criterion, that a theory own up to—and as far as possible repay—its intellectual debts, is in a way no more than standard theoretical hygiene. But it is important to highlight, in the computational case, for two intertwined reasons. First, it turns out that several candidate theories of computing (including the official "Theory of Computation" mentioned above), as well as many of the reigning but largely tacit ideas about computing held in surrounding disciplines, implicitly rely, without explanation, on such substantial, recalcitrant notions as interpretation,[7] representation, and semantics.[8] Second, which only makes matters worse, there is a widespread tendency in the surrounding intellectual terrain to point to computation as a possible *theory of those very recalcitrant notions*. Unless we ferret out all such dependencies, and lay them in plain view, we run at least two serious risks: (i) of endorsing accounts that are either based on, or give rise to, vicious conceptual circularity; and (ii) of promulgating and legitimating various unwarranted preconceptions or parochial (e.g., modernist) biases (such as of a strict mind-body dualism).

The third, "cognitive" criterion—that an adequate theory of computation provide an intelligible foundation for a theory of mind—is of a

somewhat different character. Like the second, it is more a metatheoretic requirement on the form of a theory than a constraint on its substantive content. But its elevation to a primary criterion is nonstandard, and needs explaining. Its inclusion is not based simply on the fact that the computational theory of mind (the idea that we, too, might be computers) is one of the most provocative and ramifying ideas in intellectual history, underwriting artificial intelligence, cognitive psychology, and contemporary philosophy of mind. Some other ideas about computing are just as sweeping in scope (such as proposals to unify the foundations of quantum mechanics with the foundations of information), but have not spawned their own methodological criteria here. Rather, what distinguishes the computational theory of mind, in the present context, has to do with the epistemological consequences that would follow, if it were true.

Theorizing is undeniably a cognitive endeavor. If the computational theory of mind were correct, therefore, a theory of computation would be *reflexive*—applying not only (at the object-level) to computing in general, but also (at the metalevel) to the process of theorizing. That is, the theory's claims about the nature of computing would apply to the theory itself. On pain of contradiction, therefore, unless one determines the reflexive implications of any candidate theory (of computing) on the form that the theory itself should take, and assesses the theory from such a reflexively consistent position, one will not be able to judge whether it is correct.[9]

More specifically, suppose that mind is in fact computational, and that we were to judge a candidate (object-level) theory of computing from the perspective of an implicit metatheory inconsistent with that candidate theory. And then suppose that, when judged from that perspective, the candidate theory is determined to be good or bad. There would be no reason to trust such a conclusion. For the conclusion might be due not to the empirical adequacy or failings of the theory under consideration, but rather to the conceptual inadequacy of the presumed metatheory.[10]

In sum, the plausibility of the computational theory of mind requires that a proper analysis of a candidate theory of computing must consider: (i) what computational theory of mind would be generated, in its terms; (ii) what form theories in general would take, on such a model of mind; (iii) what the candidate theory of computing in question would look like, when framed as such a theory; (iv) whether the resulting theory

(of computing), so framed, would hold true of computation-in-the-wild; and (v) whether, if it did turn out to be true (i.e., empirically), mentation and theorizing would, by those lights, also be computational. *All this is required, for reflexive integrity*. To do these things, we need to understand whether—and how—the theory could underwrite a theory of mind. Hence the cognitive criterion.

It is essential to understand, however, that the cognitive criterion requires only that we *understand* what form a computational theory of mind would take; it does not reflect any commitment to *accept* such a theory. In committing myself to honor the criterion, that is, I make no advance commitment to computationalism's being true or false. I just want to know what it says.

None of this is to say that the content of the computational theory of mind is left open. Computationalism's fundamental thesis—that the mind is computational—is given substance by the first, empirical criterion. Computationalism, that is—at least as I read it—is not a theory-laden or "opaque" proposal, in the sense of framing or resting on a specific hypothesis about what computers are. Rather, it has more an ostensive or "transparent" character: it claims that people (i.e., us) are computers in whatever way that computers (i.e., those things over there) are computers, or at least in whatever way *some* of those things are computers.[11]

It follows that specific theoretical formulations of computationalism (whether pro or con) are doubly contingent. Thus consider, on the positive side, Newell and Simon's popular (1976) "physical symbol system hypothesis," according to which human intelligence is claimed to consist in physical symbol manipulation; or Fodor's (1975, 1980) claim that thinking consists in formal symbol manipulation; or—on the critical side—Dreyfus's (1992) assertion that computationalism (as opposed to connectionism) requires the explicit manipulation of explicit symbols; or van Gelder's (1995) claim that computationalism is both false and misleading, deserving to be replaced by dynamical alternatives. Not only do all these writers make hypothetical statements about *people,* that they are or are not physical, formal, or explicit symbol manipulators, respectively; they do so by making (hypothetical) statements about *computers,* that they are in some essential or illuminating way characterizable in the same way. Because I take the latter claims to be as subservient to empirical adequacy as the former, there are two ways in which these writers could

be wrong. In claiming that people are formal symbol manipulators, for example, Fodor would naturally be wrong if computers were formal symbol manipulators and people were not. But he would also be wrong, *while the computational theory of mind itself might still be true,* if computers were not formal symbol manipulators, either. Similarly, van Gelder's brief against computational theories of mind is vulnerable to his understanding of what computing is actually like. If, as I believe, computation-in-the-wild is not as he characterizes it, then the sting of his critique is entirely eliminated.

In sum, computational cognitive science is, like computer science, hostage to the foundational project:[12] of formulating a comprehensive, true, and intellectually satisfying theory of computing that honors all three criteria.

No one of them is easy to meet.

3 Seven Construals of Computation

Some will argue that we already know what computation is. That in turn breaks into two questions: (i) is there a story—an account that people think answers the question of what computing is (computers are); and (ii) is that story right?

Regarding the first question, the answer is not *no,* but it is not a simple *yes,* either. More than one idea is at play in current theoretic discourse. Over the years, I have found it convenient to distinguish seven primary *construals* of computation, each requiring its own analysis:

1. *Formal symbol manipulation (FSM):* the idea, derivative from a century's work in formal logic and metamathematics, of a machine manipulating symbolic or (at least potentially) meaningful expressions without regard to their interpretation or semantic content;
2. *Effective computability (EC):* what can be done, and how hard it is to do it, mechanically, as it were, by an abstract analogue of a "mere machine";
3. *Execution of an algorithm (ALG) or rule-following (RF):* what is involved, and what behavior is thereby produced, in following a set of rules or instructions, such as when making dessert;
4. *Calculation of a function (FUN):* the behavior, when given as input an argument to a mathematical function, of producing as output the value of that function applied to that argument;

5. *Digital state machine (DSM):* the idea of an automaton with a finite, disjoint set of internally homogeneous machine states—as parodied in the "clunk, clunk, clunk" gait of a 1950s cartoon robot;
6. *Information processing (IP):* what is involved in storing, manipulating, displaying, and otherwise trafficking in information, whatever information might be; and
7. *Physical symbol systems (PSS):* the idea, made famous by Newell and Simon (1976), that, somehow or other, computers interact with, and perhaps also are made of, symbols in a way that depends on their mutual physical embodiment.

These seven construals have formed the core of our thinking about computation over the last fifty years, but I make no claim that this list is exhaustive.[13] At least to date, however, it is these seven that have shouldered the lion's share of responsibility for framing the intellectual debate.

By far the most important step in getting to the heart of the foundational question, I believe, is to recognize that these seven construals are all conceptually distinct. In part because of their great familiarity (we have long since lost our innocence), and in part because "real" computers seem to exemplify more than one of them—including those often-imagined but seldom-seen Turing machines, complete with controllers, read-write heads, and indefinitely long tapes—it is sometimes uncritically thought that all seven can be viewed as rough synonyms, as if they were different ways of getting at the same thing. Indeed, this conflationary tendency is rampant in the literature, much of which moves around among them as if doing so were intellectually free. But that is a mistake. The supposition that any two of these construals amount to the same thing, let alone that all seven do, is simply false.

For example, consider the formal symbol manipulation construal (FSM). It explicitly characterizes computing in terms of a semantic or intentional aspect, if for no other reason than that without some such intentional character there would be no warrant in calling it *symbol* manipulation.[14] In contrast, the digital state machine construal (DSM) makes no such reference to intentional properties. If a Lincoln-log contraption were digital but not symbolic, and a system manipulating continuous symbols were formal but not digital, they would be differentially counted as computational by the two construals. Not only do FSM and DSM *mean* different things, in other words; they (at least plausibly) have overlapping but distinct extensions.

The effective computability (EC) and algorithm execution (ALG) construals similarly differ on the crucial issue of semantics. Whereas the effective computability construal, at least in the hands of computer scientists, seems free of intentional connotation,[15] the idea of algorithm execution, at least as I have characterized it, seems not only to involve rules or recipes, which presumably do mean something, but also (pace Wittgenstein) to require some sort of understanding on the part of the agent producing the behavior.

Semantics is not the only open issue; there is also an issue of abstractness versus concreteness. For example, it is unclear whether the notions of "machine" and "taking an effective step" internal to the EC construal make fundamental reference to causal powers, material realization, or other concrete physical properties, or whether, as most current theoretical discussions suggest, effective computability should be taken as an entirely abstract mathematical notion. Again, if we do not understand this crucial aspect of the "mind-body problem for machines," how can we expect computational metaphors to help us in the case of people?

There are still other differences among construals. They differ on whether they inherently focus on internal structure or external input/output, for example—that is, on whether: (i) they treat computation as fundamentally *a way of being structured or constituted,* so that surface or externally observable behavior is derivative; or whether (ii) the *having of a particular behavior* is the essential locus of being computational, with questions about how that is achieved left unspecified and uncared about. The formal symbol manipulation and digital state machine construals are of the former, structurally constitutional sort; effective computability is of the latter, behavioral variety; algorithm execution appears to lie somewhere in the middle.

The construals also differ in the degree of attention and allegiance they have garnered in different disciplines. Formal symbol manipulation (FSM) has for many years been the conception of computing that is privileged in artificial intelligence and philosophy of mind, but it receives almost no attention in computer science. Theoretical computer science focuses primarily on the effective computability (EC) and algorithm (ALG) construals, whereas mathematicians, logicians, and most philosophers of logic and mathematics pay primary allegiance to the functional conception (FUN). Publicly, in contrast, it is surely the information pro-

cessing (IP) construal that receives the major focus—being by far the most likely characterization of computation to appear in the *Wall Street Journal,* and the idea responsible for such popular slogans as "the information age" and "the information highway."

Not only must the seven construals be distinguished one from another; additional distinctions must be made within each one. Thus the idea of information processing (IP) needs to be broken down, in turn, into at least three subreadings, depending on how "information" is understood: (i) as a *lay* notion, dating from perhaps the nineteenth century, of something like an abstract, publicly accessible commodity, carrying a certain degree of autonomous authority; (ii) so-called information theory, an at least seemingly semantics-free notion that originated with Shannon and Weaver (1949), spread out through much of cybernetics and communication theory, is implicated in Kolmogorov, Chaitin, and similar complexity measures, and has more recently been tied to notions of energy and, particularly, entropy; and (iii) the semantical notion of information advocated by Dretske (1981), Barwise and Perry (1983), Halpern (1987), and others, which in contrast to the second deals explicitly with semantic content and veridicality.

Clarifying all these issues, bringing the salient assumptions to the fore, showing where they agree and where they differ, tracing the roles they have played in the last fifty years—questions like this must be part of any foundational reconstruction. But in a sense these issues are all secondary. For none has the bite of the second question raised at the beginning of the section, namely, of whether any of the enumerated accounts is *right*.

Naturally, one has to say just what this question means—has to answer the question "Right of what?"—in order to avoid the superficial response: "Of course such and such a construal is right; that's how computation is *defined!*" This is where the empirical criterion takes hold. More seriously, I am prepared to argue for a much more radical conclusion, which we can dub as the first major result:[16]

C1. When subjected to the empirical demands of practice and the (reflexively mandated) conceptual demands of cognitive science, *all seven primary construals fail*—for deep, overlapping, but distinct, reasons.

4 Diagnosis I: General

What is the problem? Why do these theories all fail?

The answers are found at many levels. In the next section, I discuss some construal-specific problems. But a general thing can be said first. Throughout, the most profound difficulties have to do with semantics. It is widely (if tacitly) recognized that computation is in one way or another a symbolic or representational or information-based or semantical—that is, as philosophers would say, an *intentional*—phenomenon.[17] Somehow or other, though in ways we do not yet understand, the states of a computer can model or simulate or represent or stand for or carry information about or signify other states in the world (or at least can be taken by people to do so). This semantical or intentional character of computation is betrayed by such phrases as *symbol* manipulation, *information* processing, programming *languages, knowledge representation, data* bases, and so on. Indeed, if computing were not intentional, it would be spectacular that so many intentional words of English systematically serve as technical terms in computer science.[18] Furthermore—and this is important to understand—it is the intentionality of the computational that motivates the cognitivist hypothesis. The only compelling reason to suppose that we (or minds or intelligence) might be computers stems from the fact that we, too, deal with representations, symbols, meaning, information, and the like.[19]

For someone with cognitivist leanings, therefore—as opposed, say, to an eliminativist materialist, or to some types of connectionist—it is natural to expect that a comprehensive theory of computation will have to focus on its semantical aspects. This raises problems enough. Consider just the issue of representation. To meet the first criterion, of empirical adequacy, a successful candidate will have to make sense of the myriad kinds of representation that saturate real-world systems—from bit maps and images to knowledge representations and databases; from high-speed caches to long-term backup tapes; from low-level finite-element models used in simulation to high-level analytic descriptions supporting reasoning and inference; from text to graphics to audio to video to virtual reality. As well as being vast in scope, it will also have to combine decisive theoretical bite with exquisite resolution, in order to distinguish: models

from implementations; analyses from simulations; and virtual machines at one level of abstraction from virtual machines at another level of abstraction, in terms of which the former may be implemented.

To meet the second, conceptual criterion, moreover, any account of this profusion of representational practice must be grounded on, or at least defined in terms of, a theory of semantics or content, partly in order for the concomitant psychological theory to avoid vacuity or circularity, and partly so that even the computational part of the theory meet a minimal kind of naturalistic criterion: that we understand how computation is part of the natural world. This is made all the more difficult by the fact that the word "semantics" is used in an incredible variety of senses across the range of the intentional sciences. Indeed, in my experience it is virtually impossible, from any one location within that range, to understand the full significance of the term, so disparate is that practice *in toto*.

Genuine theories of content,[20] moreover—of what it is that makes a given symbol or structure or patch of the world be *about* or *oriented toward* some other entity or structure or patch—are notoriously hard to come by.[21] Some putatively foundational construals of computation are implicitly defined in terms of just such a background theory of semantics, but neither explain what semantics is, nor admit that semantical dependence—and thus fail the second, conceptual criterion. This includes the first, formal symbol manipulation construal so favored (and disparaged!) in the cognitive sciences, in spite of its superficial formulation as being "independent of semantics."[22] Other construals, such as those that view computation as the behavior of discrete automata—and also, I will argue below, even if this is far from immediately evident, the recursion-theoretic one that describes such behavior as the calculation of effective functions—fail to deal with computation's semantical aspect at all, in spite of sometimes using semantical vocabulary, and so fail the first, empirical criterion. In the end, one is driven inexorably to a second major conclusion:[23]

C2. In spite of the advance press, especially from cognitivist quarters, computer science, far from supplying the answers to fundamental intentional mysteries, must, like cognitive science, await the development of a satisfying theory of semantics and intentionality.

5 Diagnosis II: Specific

So none of the seven construals provides an account of semantics. Since I take computation to be semantic (not just by assumption, but as an unavoidable lesson from empirical investigation), that means they fail as theories of computation, as well (i.e., C2 implies C1). And that is just the beginning of the problems. All seven also fail for detailed structural reasons—different reasons per construal, but reasons that add up, overall, to a remarkably coherent overall picture.

In this section I summarize just a few of the problems, to convey a flavor of what is going on. In each case, to put this in context, these results emerge from a general effort, in the main investigation, to explicate, for each construal:

1. What the construal says or comes to—what claim it makes about what it is to be a computer;
2. Where it derives from, historically;
3. Why it has been held;
4. What is right about it—what insights it gets at;
5. What is wrong with it, conceptually, empirically, and explanatorily;
6. Why it must ultimately be replaced; and
7. What about it should nevertheless be retained in a "successor," more adequate account.

5.1 Formal Symbol Manipulation

The FSM construal has a distinctly *antisemantical* flavor, owing to its claim that computation is the "manipulation of symbols independent of their semantics." On analysis, it turns out to be motivated by two entirely different, ultimately incompatible, independence intuitions. The first motivation is at the level of the theory, and is reminiscent of a reductionist desire for a "semantics-free" account. It takes the FSM thesis to be a claim that computation can be *described* or *analyzed* in a semantics-free way. If that were true, so the argument goes, that would go some distance toward naturalizing intentionality. (As Haugeland says, ". . . if you take care of the syntax, the semantics will take care of itself" [1981a, p. 23]; see also Haugeland [1985].)

There is a second motivating intuition, different in character, that holds at the level of the phenomenon. Here the idea is simply the familiar obser-

vation that intentional phenomena, such as reasoning, hoping, or dreaming, carry on in relative independence of their subject matters or referents. Reference and truth, it is recognized, are just not the sorts of properties that can play a causal role in engendering behavior—essentially because they involve some sort of relational coordination with things that are *too far away* (in some relevant respect) to make a difference. This relational characteristic of intentionality—something I call semantic *disconnection*—is such a deep aspect of intentional phenomena that it is hard to imagine its being false. Without it, falsity would cease to exist, but so too would hypotheticals; fantasy lives would be metaphysically banned; you would not be able to think about continental drift without bringing the tectonic plates along with you.

For discussion, I label the two readings of the FSM construal *conceptual* and *ontological,* respectively.[24] The ontological reading is natural, familiar, and based on a deep insight. But it is too narrow. Many counterexamples can be cited against it, though space does not permit rehearsing them here.[25] Instead, to get to the heart of the matter, it helps to highlight a distinction between two kinds of "boundary" thought to be relevant or essential—indeed, often assumed a priori—in the analysis of computers and other intentional systems:

1. *Physical:* A physical boundary between the system and its surrounding environment—between "inside" and "outside"; and
2. *Semantic:* A semantic "boundary" between symbols and their referents.

In terms of these two distinctions, the ontological reading of the FSM construal can be understood as presuming the following two theses:

1. *Alignment:* That the physical and semantic boundaries line up, with all the symbols inside, all the referents outside; and
2. *Isolation:* That this allegedly aligned boundary is a barrier or gulf across which various forms of dependence (causal, logical, explanatory) do not reach.

The fundamental idea underlying the FSM thesis, that is, is that a barrier of this double allegedly aligned sort can be drawn around a computer, separating a pristine inner world of symbols—a private kingdom of ratiocination or thought, as it were—understood both to work (ontologically) and to be analyzable (theoretically) in isolation, without distracting influence from the messy, unpredictable exterior.

It turns out, in a way that is ultimately not surprising, that the traditional examples motivating the FSM construal, such as theorem proving in formal logic, meet this complex pair of conditions. First, they involve internal symbols designating external situations, thereby satisfying *Alignment:* (internal) databases representing (external) employee salaries, (internal) differential equations modeling the (external) perihelion of Mercury, (internal) first-order axioms designating (external) Platonic numbers or purely abstract sets, and so on. Second, especially in the paradigmatic examples of formal axiomatizations of arithmetic and proof systems of first-order logic (and, even more especially, when those systems are understood in classical, especially model-theoretic guise), the system is assumed to exhibit the requisite lack of interaction between the (internal) syntactic proof system and the (external, perhaps model-theoretic) interpretation, satisfying *Isolation*. In conjunction, the two assumptions allow the familiar two-part picture of a formal system to emerge: of a locally contained syntactic system, on the one hand, consisting in symbols or formulae in close causal intimacy with a proof-theoretic inference regimen; and a remote realm of numbers or sets or "ur-elements," in which the symbols or formulae are interpreted, on the other. It is because the formality condition relies on both theses together that the classical picture takes computation to consist exclusively in symbol-symbol transformations, carried on entirely within the confines of a machine.

The first—and easier—challenge to the antisemantical thesis comes when one retains the first *Alignment* assumption, of coincident boundaries, but relaxes the second *Isolation* claim, of no interaction. This is the classical realm of input/output, home of the familiar notion of a transducer. And it is here that one encounters the most familiar challenges to the FSM construal, such as the "robotic" and "system" replies to Searle's (1980) Chinese room argument, and Harnad's (1990) "Total Turing Test" as a measure of intelligence. Thus imagine a traditional perception system—for example, one that on encounter with a mountain lion constructs a symbolic representation of the form *mountain-lion-043*. There is interaction (and dependence) from external world to internal representation. By the same token, an actuator system, such as one that would allow a robot to respond to a symbol of the form *cross-the-street* by

moving from one side of the road to the other, violates the *Isolation* assumption in the other direction, from internal representation to external world.

Note, in spite of this interaction, and the consequent violation of *Isolation,* that *Alignment* is preserved in both cases: the transducer is imagined to mediate between an internal symbol and an external referent. Nevertheless, the violation of *Isolation* is already enough to defeat the formality condition. This is why transducers and computation are widely recognized to be uneasy bedfellows, at least when formality is at issue. It is also why, if one rests the critique at this point, defenders of the antisemantical construal are tempted to wonder, given that the operations of transducers violate *formality,* whether they should perhaps be counted as *not being computational.*[26] Given the increasing role of environmental interaction within computational practice, it is not at all clear that this would be possible, without violating the condition of empirical adequacy embraced at the outset. For our purposes it does not ultimately matter, however, because the critique is only halfway done.

More devastating to the FSM construal are examples that challenge the *Alignment* thesis. It turns out, on analysis, that far from lining up on top of each other, real-world computer systems' physical and semantic boundaries *cross-cut,* in rich and productive interplay. It is not just that computers are involved in an engaged, participatory way with *external* subject matters, in other words, as suggested by some recent "situated" theorists. They are participatorily engaged in the world *as a whole*—in a world that indiscriminately includes themselves, their own internal states and processes. This integrated participatory involvement, blind to any a priori subject-world distinction, and concomitantly intentionally directed toward both internally and externally exemplified states of affairs, is not only architecturally essential, but is also critical, when the time comes, in establishing and grounding a system's intentional capacities.

From a purely structural point of view, four types of case are required to demonstrate this nonalignment of boundaries: (i) where a symbol and referent are both internal; (ii) where a symbol is internal and its referent external; (iii) where symbol and referent are both external; and (iv) where symbol is external and its referent internal. The first is exemplified in

cases of quotation, metastructural designation, window systems, e-mail, compilers, loaders, network routers, and at least arguably all programs (as opposed, say, to databases). The second, of internal symbols with external referents, can be considered as something of a theoretical (though not necessarily empirical) default, as for example when one reflects on the sun's setting over Georgian Bay (to use a human example), or when a computer database represents the usage pattern of a set of university classrooms. The third and fourth are neither more nor less than a description of ordinary written text, public writing, and so on—to say nothing of pictures, sketches, conversations, and the whole panoply of other forms of external representation. Relative to any particular system, they are distinguished by whether the subject matters of those external representations are similarly external, or are internal. The familiar red skull-and-crossbones signifying radioactivity is external to both man and machine and also denotes something external to man and machine, and thus belongs to the third category. To a computer or person involved, on the other hand, an account of how they work (psychoanalysis of person or machine, as it were, to say nothing of logic diagrams, instruction manuals, etc.) is an example of the fourth.

By itself, violating *Alignment* is not enough to defeat formality. What it does accomplish, however, is to radically undermine *Isolation*'s plausibility. In particular, the antisemantical thesis constitutive of the FSM construal is challenged not only because these examples show that the physical and semantic boundaries cross-cut, thereby undermining the *Alignment* assumption, but because they illustrate the presence, indeed the prevalence, of effective traffic across both boundaries—between and among all the various categories in question—thereby negating *Isolation.*

And this negation of *Isolation,* in turn, shows up, for what it is, the common suggestion that transducers, because of violating the antisemantical thesis, should be ruled "out of court"—should be taken as noncomputational, à la Devitt (1991).[27] It should be clear that this maneuver is ill advised; it's even a bit of a cop-out. For consider what a proponent of such a move must face up to, when confronted with boundary nonalignment. *The notion of a transducer must be split in two.* In order to retain an antisemantical (FSM) construal of computing, someone interested in transducers would have to distinguish:

1. *Physical transducers,* for operations or modules that cross or mediate between the inside and outside of a system; and
2. *Semantic transducers,* for operations or modules that mediate or "cross" between symbols and their referents.

And it is this bifurcation, finally, that irrevocably defeats the antisemantical formalists' claim. For the only remotely plausible notion of transducer, in practice, is the physical one. That is what we think of when we imagine vision, touch, smell, articulation, wheels, muscles, and the like: systems that mediate between the internals of a system and the "outside" world. Transducers, that is, at least in informal imagination of practitioners, are for connecting systems to their (physical) environments.[28] What poses a challenge to the formal (antisemantical) symbol manipulation construal of computation, on the other hand, are *semantic* transducers: those aspects of a system that involve trading between occurrent states of affairs, on the one hand, and representations of them, on the other. Antisemantics is challenged as much by disquotation as by driving around.

As a result, the only way to retain the ontological version of the FSM construal is to disallow (i.e., count as noncomputational) the operations of semantic transducers. *But that is absurd*. It makes it clear, ultimately, that distinguishing that subset of computation which satisfies the ontological version of the antisemantical claim is not only unmotivated, solving the problem by fiat (making it uninteresting), but is a spectacularly infeasible way to draw and quarter any actual, real-life system. For no one who has ever built a computational system has ever found any reason to bracket reference-crossing operations, or to treat them as a distinct type. Not only that; think of how many different kinds of examples of semantic transducer one can imagine: counting, array indexing, e-mail, disquotation, error-correction circuits, linkers, loaders, simple instructions, database access routines, pointers, reflection principles in logic, index operations into matrices, most Lisp primitives, and the like. Furthermore, to *define* a species of transducer in this semantical way, and then to remove them from consideration as not being genuinely computational, would make computation (minus the transducers) antisemantical *tautologically*. It would no longer be an interesting claim about the world that computation was antisemantical—an insight into how things are. Instead, the word "computation" would simply be shorthand for

antisemantical symbol manipulation. The question would be whether anything interesting was in this named class—and, in particular, whether this conception of computation captured the essential regularities underlying practice. And we have already seen the answer to that: it is *no*.

In sum, introducing a notion of a semantical transducer solves the problem tautologically, cuts the subject matter at an unnatural joint, and fails to reconstruct practice. That is quite a lot to have going against it.

Furthermore, to up the ante on the whole investigation, not only are these cases of "semantic transduction" all perfectly well behaved; they even seem, intuitively, to be as "formal" as any other kind of operation. If that is so, then those systems either are not formal, after all, *or else the word "formal" has never meant independence of syntax and semantics in the way that the FSM construal claims*. Either way, the ontological construal does not survive.

Though it has been framed negatively, we can summarize this result in positive terms:

C3. Rather than consisting of an internal world of symbols separated from an external realm of referents, as imagined in the FSM construal, real-world computational processes are *participatory,* in the following sense: they involve complex paths of causal interaction between and among symbols and referents, both internal and external, cross-coupled in complex configurations.

5.2 Effective Computability

Although different in detail, the arguments against the other major construals have a certain similarity in style. In each case, the strategy in the main investigation has been to develop a staged series of counterexamples, not simply to show that the construal is false, but to serve as strong enough intuition pumps on which to base a positive alternative. In other words, the point is not critique, but deconstruction en route to reconstruction. Space permits a few words about just one other construal: effective computability—the idea that underwrites recursion theory, complexity theory, and, as I have said, the official (mathematical) "Theory of Computation."

Note, for starters—as mentioned earlier—that whereas the first, FSM construal is predominant in artificial intelligence, cognitive science, and

philosophy of mind, it is the second, effective computability (EC) construal, in contrast, that underlies most theoretical and practical computer science.

Fundamentally, it is widely agreed, the theory of effective computability focuses on "what can be done by a mechanism." But two conceptual problems have clouded its proper appreciation. First, in spite of its subject matter, it is almost always characterized *abstractly,* as if it were a branch of mathematics. Second, it is imagined to be a theory defined over (for example) the numbers. Specifically, the marks on the tape of the paradigmatic Turing machine are viewed as *representations* or *encodings*—representations, in general, or at least in the first instance, of numbers, functions, or other Turing machines.

In almost exact contrast to the received view, I argue two things. First, I claim that the theory of effective computability is fundamentally a theory about the *physical* nature of patches of the world. In underlying character, I believe, it is no more "mathematical" than anything else in physics—even if we use mathematical structures to model that physical reality. Second—and this is sure to be contentious—I argue that recursion theory is fundamentally a *theory of marks.* More specifically, rather than taking the marks on the tape to be representations of numbers, as has universally been assumed in the theoretical tradition, I defend the following claim:

C4. The representation relation for Turing machines, alleged to run from marks to numbers, in fact runs the other way, from numbers to marks. The truth is 180° off what we have all been led to believe.

In the detailed analysis various kinds of evidence are cited in defense of this nonstandard claim. For example:

1. Unless one understands it this way, one can solve the halting problem;[29]
2. An analysis of history, through Turing's paper and subsequent work, especially including the development of the universal Turing machine, shows how and why the representation relation was inadvertently turned upside down in this way;
3. The analysis makes sense of a number of otherwise-inexplicable practices, including, among other examples: (i) the use of the word "semantics" in practicing computer science to signify the behavior engendered by running a program,[30] (ii) the rising popularity of such conceptual tools

as Girard's linear logic, and (iii) the close association between theoretical computer science and constructive mathematics.

It follows from this analysis that all use of semantical vocabulary in the "official" Theory of Computation is metatheoretic. As a result, *the so-called (mathematical) "Theory of Computation" is not a theory of intentional phenomena*—in the sense that it is not a theory that deals with its subject matter *as* an intentional phenomenon.

In this way the layers of irony multiply. Whereas the FSM construal fails to meet its own criterion, of being "defined independent of semantics," this second construal *does* meet (at least the conceptual reading of) that first-construal condition. Exactly in achieving that success, however, the recursion-theoretic tradition thereby fails. For computation, as was said above, and as I am prepared to argue, *is* (empirically) an intentional phenomenon. So the EC construal achieves naturalistic palatability at the expense of being about the wrong subject matter.

We are thus led inexorably to the following strong conclusion: that what goes by the name "Theory of Computation" fails not because it makes false claims about computation, but because *it is not a theory of computation at all*.[31,32]

In sum, the longer analysis ultimately leads to a recommendation that we redraw a substantial portion of our intellectual map. What has been (indeed, by most people still is) called a "Theory of Computation" is in fact a general theory of the physical world—specifically, a theory of how hard it is, and what is required, for patches of the world in one physical configuration to change into another physical configuration. It applies to *all* physical entities, not just to computers. It is no more mathematical than the rest of physics, in using (abstract) mathematical structures to model (concrete) physical phenomena. Ultimately, therefore, it should be joined with physics—because in a sense it *is* physics.

We can put this result more positively. Though falsely (and misleadingly) labeled, the mathematical Theory of Computation has been a spectacular achievement, of which the twentieth century should be proud. Indeed, this is important enough that we can label it as the fifth major result:

C5. Though not yet so recognized, the mathematical theory based on recursion theory, Turing machines, complexity analyses, and the like—

widely known as the "Theory of Computation"—is neither more nor less than a *mathematical theory of the flow of causality*.

6 Method

Similarly strong conclusions can be arrived at by pursuing each of the other construals. Indeed, the conclusion from the analysis of the digital state machine construal (DSM)—that computation-in-the-wild is not digital—is, if anything, even more consequential than the results derived from either the FSM or the EC critiques. Rather than go into more construals here, however, I instead want to say a word about method—specifically, about *formality*. For a potent theme underlies all seven critiques: that part of what has blinded us to the true nature of computation has to do with the often pretheoretic assumption that *computation and/or computers are formal*.

In one way or another, no matter what construal they pledge allegiance to, just about everyone thinks that computers are formal—that they manipulate symbols formally, that programs (formally) specify formal procedures, that data structures are a kind of formalism, that computational phenomena are uniquely suited for analysis by formal methods, and so on. In fact, the computer is often viewed as the crowning achievement of an entire "formal tradition"—an intellectual orientation, reaching back through Galileo to Plato, that was epitomized in the twentieth century in the logic and metamathematics of Frege, Russell, Whitehead, Carnap, and Turing, among others.

This history would suggest that formality is an essential aspect of computation. But since the outset, I have not believed that this is necessarily so. For one thing, it has never been clear what the allegiance to formality is an allegiance to. It is not as if "formal" is a technical or theory-internal predicate, after all. People may believe that developing an idea means formalizing it, and that programming languages are formal languages, and that theorem provers operate on formal axioms—but few write "*formal*(x)" in their daily equations. Moreover, a raft of different meanings and connotations of this problematic term lies just below the surface. Far from hurting, this apparent ambiguity has helped to cement popular consensus. Freed of the need to be strictly defined ("*formal*" is not a formal predicate), formality has been able to serve as a lightning rod for

a cluster of ontological assumptions, methodological commitments, and social and historical biases.

Because it remains tacit, cuts deep, has important historical roots, and permeates practice, formality has been an ideal foil, over the years, in terms of which to investigate computation.

Almost a dozen different readings of "formal" can be gleaned from informal usage: *precise, abstract, syntactic, mathematical, explicit, digital, a-contextual, nonsemantic,* among others.[33] They are alike in foisting recalcitrant theoretical issues onto center stage. Consider explicitness, for example, of the sort that might explain such a sentence as "for theoretical purposes we should lay out our tacit assumptions in a formal representation." Not only have implicitness and explicitness stubbornly resisted theoretical analysis, but both notions are parasitic on something else we do not understand: general representation.[34] Or consider "a-contextual." Where is an overall theory of context in terms of which we can understand what it would be to say of something (a logical representation, say) that it was not contextually dependent?

Considerations like this suggest that particular readings of formality can be most helpfully pursued within the context of the general theoretical edifices that have been constructed (more or less explicitly) in their terms. Five are particularly important:

1. The *antisemantical* reading mentioned above: the idea that a symbolic structure (representation, language, program, symbol system, etc.) is formal just in case it is manipulated *independent of its semantics*. Paradigmatic cases include so-called formal logic, in which it is assumed that a theorem—such as *Mortal*(*Socrates*)—is derived by an automatic inference regimen without regard to the reference, truth, or even meaning of any of its premises.
2. A closely allied grammatical or *syntactic* reading, illustrated in such a sentence as "inference rules are defined in terms of the *formal* properties of expressions." (Note that whereas the antisemantical reading is negatively characterized, this syntactic one has a positive sense.)
3. A reading meaning something like *determinate* or *well-defined*—that is, as ruling out all ambiguity and vagueness. This construal turns out to be related to a variant of the computationally familiar notion of digitality or discreteness.
4. A construal of "formal" as essentially equivalent to *mathematical.*
5. A reading that cross-cuts the other four: formality as applied to analyses or *methods,* perhaps with a derivative ontological implication that

some subject matters (including computation?) are uniquely suited to such analytic techniques.

The first two (antisemantical and syntactic) are often treated as conceptually equivalent, but to do that is to assume that a system's syntactic and semantic properties are *necessarily disjoint*—which is almost certainly false. The relationship between the third (determinate) reading and digitality does not have so much to do with what Haugeland (1982) calls "first-order digitality": the ordinary assumption that a system's states can be partitioned into a determinate set, such that its future behavior or essence stems solely from membership in one element of that set, without any ambiguity or matter of degree. Rather, vagueness and indefiniteness (as opposed to simple continuity) are excluded by a *second-order* form of digitality—digitality at the level of concept or type, in the sense of there being a binary "yes/no" fact of the matter about whether any given situation falls under (or is correctly classified in terms of) the given concept. And finally, the fourth view—that to be formal has something to do with being mathematical, or at least with being mathematically characterizable—occupies something of an ontological middle realm between the subject-matter orientation of the first three and the methodological orientation of the fifth.

The ultimate moral for computer and cognitive science, I argue, is similar to the claim made earlier about the seven construals: *not one of these readings of "formal" correctly applies to the computational case.* It can never be absolutely proved that computation is not formal, of course, given that the notion of formality is not determinately tied down. What I am prepared to argue (and do argue in the full analysis) is the following: no standard construal of formality, including any of those enumerated above, is both (i) substantive and (ii) true of extant computational practice. Some readings reduce to vacuity, or to no more than physical realizability; others break down in internal contradiction; others survive the test of being substantial, but are demonstrably false of current systems. In the end, one is forced to a sixth major conclusion:

C6. Computation is not formal.

It is an incredible historical irony: the computer, darling child of the formal tradition, has outstripped the bounds of the very tradition that gave rise to it.

7 The Ontological Wall

Where does all this leave us? It begins to change the character of the project. It is perhaps best described in personal terms. Over time, investigations of the sort described above, and consideration of the conclusions reached through them, convinced me that none of the reigning theories or construals of computation, nor any of the reigning methodological attitudes toward computation, will *ever* lead to an analysis strong enough to meet the three criteria laid down at the outset.

It wasn't always that way. For the first twenty years of the investigation I remained:

1. in awe of the depth, texture, scope, pluck, and impact of computational practice;
2. critical of the inadequate state of the current theoretical art;
3. convinced that a formal methodological stance stood in the way of getting to the heart of the computational question; and
4. sure in my belief that what was needed, above all else, was a *non-formal*—i.e., situated, embodied, embedded, indexical, critical, reflexive, all sorts of other things (it changed, over the years)—theory of representation and semantics, in terms of which to reconstruct an adequate conception of computing.

In line with this metatheoretic attitude, as the discussion this far will have suggested, I kept semantical and representational issues in primary theoretical focus. Since, as indicated in the last section, the official "Theory of Computation," derived from recursion and complexity theory, pays no attention to such intentional problems, to strike even this much of a semantical stance was to part company with the center of gravity of the received theoretical tradition.

You might think that this would be conclusion enough. And yet, in spite of the importance and magnitude of these intentional difficulties, and in spite of the detailed conclusions suggested above, I have gradually come to believe something much more sobering: a conclusion that, although not as precisely stated as the foregoing, is if anything even more consequential:

C7. The most serious problems standing in the way of developing an adequate theory of computation are as much *ontological* as they are semantical.

It is not that computation's semantic problems go away; they remain as challenging as ever. It is that they are joined—on center stage, as it were—by even more demanding problems of ontology.

Except that to say "joined" is misleading, as if it were a matter of simple addition—as if now there were two problems on the table, whereas before there had been just one. No such luck. The two issues (representation and ontology) are inextricably entangled—a fact of obstinate theoretical and metatheoretical consequence.

A methodological consequence will illustrate the problem. Especially within the analytic tradition (by which I mean to include not just analytic philosophy, e.g., of language and mind, but most of modern science as well, complete with its formal/mathematical methods), it is traditional to analyze semantical or intentional systems, such as computers or people, under the following presupposition: (i) that one can parse or register the relevant theoretical situation in advance into a set of objects, properties, types, relations, equivalence classes, and so on (e.g., into people, heads, sentences, data structures, real-world referents, etc.)—as if this were theoretically innocuous—and then (ii), with that ontological parse in hand, go on to proclaim this or that or the other thing as an empirically justified result. Thus for example one might describe a mail-delivering robot by first describing an environment of offices, hallways, people, staircases, litter, and the like, through which the robot is supposed to navigate, and then, taking this characterization of its context as given, ask how or whether the creature represents routes, say, or offices, or the location of mail delivery stations.

If one adopts a reflexively critical point of view, however, as I have systematically been led to do (and as is mandated by the cognitive criterion), one is led inexorably to the following conclusion: that, in that allegedly innocent pretheoretical "set-up" stage, one is liable, even if unwittingly, to project so many presuppositions, biases, and advance "clues" about the "answer," and in general to so thoroughly prefigure the target situation, without either apparent or genuine justification, that *one cannot, or at least should not, take any of the subsequent "analysis" terribly seriously*. It is a general problem that I have elsewhere labeled *preemptive registration*.[35] It is problematic not just because it rejects standard analyses, but because it seems to shut all inquiry down. What else can one do, after all? How can one not parse the situation in advance

(since it will hardly do merely to whistle and walk away)? And if, undaunted, one were to go ahead and parse it anyway, what kind of story could possibly serve as a justification? It seems that any conceivable form of defense would devolve into another instance of the same problem.

In sum, the experience is less one of facing an ontological challenge than of running up against a seemingly insuperable ontological wall. Perhaps not quite of slamming into it, at least in my own case; recognition dawned slowly. But neither is the encounter exactly gentle. It is difficult to exaggerate the sense of frustration that can come, once the conceptual fog begins to clear, from seeing one's theoretical progress blocked by what seems for all the world to be an insurmountable metaphysical obstacle.

Like many of the prior claims I have made, such as that all extant theories of computation are inadequate to reconstruct practice, or that no adequate conception of computing is formal, this last claim, that theoretical progress is stymied for lack of an adequate theory of ontology, is a strong statement, in need of correspondingly strong defense. Providing that defense is one of the main goals of *AOS*. In my judgment, to make it perfectly plain, despite the progress that has been made so far, and despite the recommended adjustments reached in the course of the seven specific analyses enumerated above, we are not going to get to the heart of computation, representation, cognition, information, semantics, or intentionality, until the ontological wall is scaled, penetrated, dismantled, or in some other way defused.

One reaction to the wall might be depression. Fortunately, however, the prospects are not so bleak. For starters, there is some solace in company. It is perfectly evident, once one raises one's head from the specifically computational situation and looks around, that computer scientists, cognitive scientists, and artificial intelligence researchers are not the only ones running up against severe ontological challenges. Similar conclusions are being reported from many other quarters. The words are different, and the perspectives complementary, but the underlying phenomena are the same.

Perhaps the most obvious fellow travelers are literary critics, anthropologists, and other social theorists, vexed by what analytic categories to use in understanding people or cultures that, by such writers' own admission, comprehend and constitute the world using concepts alien to

the theorists' own. What makes the problem particularly obvious, in these cases, is the potential for *conceptual clash* between theorist's and subject's worldview—a clash that can easily seem paralyzing. One's own categories are hard to justify, and reek of imperialism; it is at best presumptuous, and at worst impossible, to try to adopt the categories of one's subjects; and it is manifestly impossible to work with no concepts at all. So it is unclear how, or even whether, to proceed.

But conceptual clash, at least outright conceptual clash, is not the only form in which the ontological problem presents itself. Consider the burgeoning interest in self-organizing and complex systems mentioned earlier, currently coalescing in a somewhat renegade subdiscipline at the intersection of dynamics, theoretical biology, and artificial life. This community debates the "emergence of organization," the units on which selection operates, the structure of self-organizing systems, the smoothness or roughness of fitness landscapes, and the like. In spite of being disciplinarily constituting, however, these discussions are conducted in the absence of adequate theories of what organization is, of what a "unit" consist in, of how "entities" arise (as opposed to how they survive), of how it is determined what predicates should figure in characterizing a fitness landscape as rough or smooth, and so on. The ontological lack is to some extent recognized in increasingly vocal calls for "theories of organization."[36] But the calls have not yet been answered.

Ontological problems have also plagued physics for years, at least since foundational issues of interpretation were thrown into relief by the developments of relativity and quantum mechanics (including the perplexing wave-particle duality, and the distinction between "classical" and "quantum" worldviews). They face connectionist psychologists, who, proud of having developed architectures that do not rely on the manipulation of formal symbol structures encoding high-level concepts, and thus of having thereby rejected propositional content, are nevertheless at a loss as to say what their architectures *do* represent. And then of course there are communities that tackle ontological questions directly: not just philosophy, but fields as far-flung as poetry and art, where attempts to get in, around, and under objects have been pursued for centuries.

So there are fellow-travelers. But no one, so far as I know, has developed an alternative ontological/metaphysical proposal in sufficient detail and depth to serve as a practicable foundational for a revitalized scientific

practice. Unlike some arguments for realism or irrealism, unlike some briefs pro or con this or that philosophy of science, and unlike as well the deliberations of science studies and other anthropological and sociological and historical treatises about science, the task I have in mind is not the increasingly common *meta*-metaphysical one—of arguing for or against a way of proceeding, if one were ever to proceed, or arguing that science proceeds in this or that way. Rather, the concrete demand is for a detailed, worked-out account—an account that "goes the distance," in terms of which accounts of particular systems can be formulated, and real-world construction proceed.

For this purpose, with respect to the job of developing an alternative metaphysics, the computational realm has unparalleled advantage. Midway between matter and mind, computation stands in excellent stead as a supply of concrete cases of middling complexity—what in computer science is called an appropriate "validation suite"—against which to test the mettle of specific metaphysical hypotheses. "Middling" in the sense of neither being so simple as to invite caricature, nor so complex as to defy comprehension. It is the development of a laboratory of this intermediate sort, halfway between the frictionless pucks and inclined planes of classical mechanics and the full-blooded richness of the human condition, that makes computing such an incredibly important stepping-stone in intellectual history.

Crucially, too, computational examples are examples with which we are as much practically as theoretically familiar (we build systems better than we understand them). Indeed—and by no means insignificantly—there are many famous divides with respect to which computing sits squarely in the middle.

8 Summary

Thus the ante is upped one more time. Not only must an adequate account of computation (any account that meets the three criteria with which we started) include a theory of semantics; it must also include a theory of ontology. Not just intentionality is at stake, in other words; so is metaphysics. But still we are not done. For on top of the foregoing strong conclusions lies an eighth one—if anything even stronger:

C8. Computation is not subject matter.

In spite of everything I said about a comprehensive, empirical, conceptually founded "theory of computing," that is, and in spite of everything I myself have thought for decades, I no longer believe that there is a distinct ontological category of computing or computation, one that will be the subject matter of a deep and explanatory and intellectually satisfying theory. Close and sustained analysis, that is, suggests that the things that Silicon Valley calls computers, the things that perforce *are* computers, do not form a coherent intellectually delimited class. Computers turn out in the end to be rather like cars: objects of inestimable social and political and economic and personal importance, but not in and of themselves, *qua* themselves, the focus of enduring scientific or intellectual inquiry—not, as philosophers would say, *natural kinds*.

Needless to say, this is another extremely strong claim—one over which some readers may be tempted to rise up in arms. At the very least, it is easy to feel massively let down, after all this work. For if I am right, it is not just that we *currently* have no satisfying intellectually productive theory of computing, of the sort I initially set out to find. Nor is it just that, through this whole analysis, I have failed to provide one. It is the even stronger conclusion that such projects will *always* fail; we will *never* have such a theory. So all the previous conclusions must be revised. It is not just that a theory of computation will not *supply* a theory of semantics, for example, as Newell has suggested; or that it will not *replace* a theory of semantics; or even that it will *depend or rest on* a theory of semantics, as was intimated at the end of section 4. It will do none of these things because *there will be no theory of computation at all*.

Given the weight that has been rested on the notion of computation—not just by me, or by computer science, or even by cognitive science, but by the vast majority of the surrounding intellectual landscape—this (like the previous conclusion about ontology) might seem like a negative result. (Among other things, you might conclude I had spent these thirty years in vain.) But in fact there is no cause for grief; for the negativity of the judgment is only superficial, and in fact almost wholly misleading. In fact I believe something almost wholly opposite, which we can label as a (final) conclusion in its own right:

C9. The superficially negative conclusion (that computing is not a subject matter) makes the twentieth-century arrival of computation onto the intellectual scene a vastly more interesting and important phenomenon than it would otherwise have been.

On reflection, it emerges that the fact that neither computing nor computation will sustain the development of a theory is by far the most exciting and triumphal conclusion that the computer and cognitive sciences could possibly hope for.

Why so? Because I am not saying that computation-in-the-wild is intrinsically atheoretical—and thus that there will be no theory of these machines, at all, when day is done. Rather, the claim is that such theory as there is—and I take it that there remains a good chance of such a thing, as much as in any domain of human activity—will not be a theory of *computation* or *computing*. It will not be a theory of computation because *computers per se,* as I have said, do not constitute a distinct, delineated subject matter. Rather, what computers are, I now believe—and what the considerable and impressive body of practice associated with them amounts to—is neither more nor less than *the full-fledged social construction*[37] *and development of intentional artifacts*. That means that the range of experience and skills and theories and results that have been developed within computer science—astoundingly complex and far-reaching, if still inadequately articulated—is best understood as practical, synthetic, raw material for no less than full theories of causation, semantics, and ontology—that is, for *metaphysics full bore*.

Where does that leave things? Substantively, it leads inexorably to the conclusion that metaphysics, ontology, epistemology, and intentionality are the only integral intellectual subject matters in the vicinity of either computer or cognitive science. Methodologically, it means that our experience with constructing computational (i.e., intentional) systems may open a window onto something to which we would not otherwise have any access: the chance to witness, with our own eyes, how intentional capacities can arise in a "merely" physical mechanism.

It is sobering, in retrospect, to realize that our *preoccupation with the fact that computers are computational* has been the major theoretical block in the way of our understanding how important computers are. They are computational, of course; that much is tautological. But only

when we *let go of the conceit that that fact is theoretically important*—only when we abandon the "c-word"—will we finally be able to see, without distraction, and thereby, perhaps, at least partially to understand, how a structured lump of clay can sit up and think.

And so that, for the last decade or so, has been my project: to take, from the ashes of computational critique, enough positive morals to serve as the inspiration, basis, and testing ground for an entirely new metaphysics. A story of subjects, a story of objects, a story of reference, a story of history.

For sheer ambition, physics does not hold a candle to computer or cognitive—or rather, as we should now call it, in order to recognize that we are dealing with something on the scale of natural science—*epistemic* or *intentional* science. Hawking (1988) and Weinberg (1994) are wrong. It is we, not the physicists, who must develop a theory of everything.

Notes

1. This chapter is distilled from, and is intended to serve as an introduction to, a series of books that collectively report, in detail, on the investigation identified in section 2. The study of computing will be presented in *The Age of Significance* (Smith, forthcoming—henceforth *AOS*); the metaphysical territory to which that study leads is introduced in *On the Origin of Objects* (Smith 1996).

2. The same thesis is sometimes referred to as *cognitivism,* though strictly speaking the term "cognitivism" denotes a more specific thesis, which takes mentation to consist in rational deliberation based on patterns of conceptualist (i.e., "cognitive") inference, reminiscent of formal logic, and usually thought to be computationally implemented (see Haugeland 1978).

3. As explained in *AOS,* the aim is to include not only the machines, devices, implementations, architectures, programs, processes, algorithms, languages, networks, interactions, behaviors, interfaces, etc., that constitute computing, but also the design, implementation, maintenance, and even use of such systems (such as Microsoft Word). Not, of course, that a theory will explain any *particular* architecture, language, etc. Rather, the point is that a foundational theory should explain *what an architecture is,* what constraints architectures must meet, etc.

4. Indeed, I ultimately argue that that theory—trafficking in Turing machines, notions of "effective computability," and the like—fails as a theory of computing, in spite of its name and its popularity. It is simultaneously too broad, in applying to more things than computers, and too narrow, in that it fails to apply to some things that are computers. More seriously, what it is a theory of, is not *computing*. See section 5.2.

5. Methodological issues arise, owing to the fact that we (at least seem to) make up the evidence. Although this ultimately has metaphysical as well as methodological implications, it undermines the empirical character of computer science no more than it does in, say, sociology or linguistics.

6. Adapted from Hutchins's *Cognition in the Wild* (1995).

7. "Interpretation" is a technical notion in computing; how it relates to the use of the term in ordinary language, or to what "interpretation" is thought to signify in literary or critical discussions, is typical of the sort of question to be addressed in the full analysis.

8. A notable example of such a far-from-innocent assumption is the widespread theoretical tendency to distinguish (i) an abstract and presumptively fundamental notion of "computation" from (ii) a concrete but derivative notion of a "computer"—the latter simply being taken to be any physical device able to carry out a computation. It turns out, on inspection, that this assumption builds in a residually dualist stance toward what is essentially the mind-body problem—a stance I eventually want to argue against, and at any rate not a thesis that should be built into a theory of computing as a presumptive but inexplicit premise.

9. For example, it would be inconsistent simultaneously to claim the following three things: (i) as many do, that scientific theories should be expressed from an entirely third-person, nonsubjective point of view; (ii) as an intrinsic fact about all computational processes, that genuine reference is possible only from a first-person, subjective vantage point ("first-person" from the perspective of the machine); and (iii) that the computational theory of mind is true. If one were to believe in the ineliminably first-person character of computational reference, and that human reference is a species of computational reference, then consistency would demand that such a theory be stated *from a first-person point of view*—since, by hypothesis, no other way of presenting the theory would refer.

10. Note that the situation is symmetric; reflexive inconsistencies can generate both false negatives and false positives.

11. The computational theory of mind does not claim that minds and computers are equivalent (in the sense that anything that is a mind is a computer, and vice versa). Rather, the idea is that minds are (at least) a *kind* of computer, and furthermore that the kind is *itself computationally characterized* (i.e., that the characteristic predicate on the restricted class of computers that are minds is itself framed in computational terms).

12. Foundationalism is widely decried, these days—especially in social and critical discourses. Attempting a foundational reconstruction of the sort I am attempting here may therefore be discredited, by some, in advance. As suggested in Smith (1996), however, I do not believe that any of the arguments that have been raised against foundationalism (particularly: against the valorization of a small set of types or categories as holding an unquestioned and/or uniquely privileged status) amounts to an argument against rigorously plumbing the depths of an intellectual subject matter. In this chapter, my use of the term "foundational"

should be taken as informal and, to an extent, lay (I am as committed as anyone to the fallacies and even dangers of master narratives, ideological inscription, and/or uniquely privileging any category or type).

13. Especially as the boundaries between computer science and surrounding intellectual territory erode (itself a development predicted by this analysis; see section 8), several ideas that originated in other fields are making their way into the center of computational theorizing as alternative conceptions of computing. At least three are important enough to be seen as construals in their own right (though the first is not usually assumed to have any direct connection with computing, and the latter two are not normally assumed to be quite as "low-level" or foundational as the primary seven):

8. *Dynamics (DYN):* the notion of a dynamical system, linear or nonlinear, as popularized in discussions of attractors, turbulence, criticality, emergence, etc.;

9. *Interactive agents (IA):* active agents enmeshed in an embedding environment, interacting and communicating with other agents (and perhaps also with people); and

10. *Self-organizing* or *complex adaptive systems (CAS):* a notion—often associated with the Santa Fe Institute—of self-organizing systems that respond to their environment by adjusting their organization or structure, so as to survive and (perhaps even) prosper.

Additional construals may need to be added, over time. Moreover, there are even those who deny that computation has *any* ontologically distinct identity. Thus Agre (1997a), for example, claims that computation should instead be methodologically individuated:

11. *Physical implementation (PHY):* a methodological hypothesis that computation is not ontologically distinct, but rather that computational practice is human expertise in the physical or material implementation of (apparently arbitrary) systems.

14. See note 22.

15. At least some logicians and philosophers, in contrast, do read the effective computability construal semantically. This difference is exactly the sort of question that needs to be disentangled and explained in the full analysis.

16. This numbering system (C1–C9) is used only for purposes of this chapter; it will not necessarily be used in *AOS*.

17. Although the term "intentional" is primarily philosophical, there are many philosophers, to say nothing of some computer and cognitive scientists, who would deny that computation is an intentional phenomenon. Reasons vary, but the most common goes something like this: (i) that computation is both *syntactic* and *formal,* where "formal" means "independent of semantics"; and (ii) that intentionality has fundamentally to do with semantics; and therefore (iii) that computation is thereby not intentional. I believe this is wrong, both empirically (that computation is purely syntactic) and conceptually (that being syntactic is a way of not being intentional); I also disagree that being intentional has *only* to do with semantics, which the denial requires. See note 22.

18. Thus computer science's use of (the English words) "language," "representation," "data," etc. is analogous to physics' use of "work," "force," "energy," etc.—as opposed to its use of "charm." That is, it reflects a commitment to do scientific justice to the center of gravity of the word's natural meaning, rather than being mere whimsical fancy.

19. Physically, we and (at least contemporary) computers are not very much alike—though it must be said that one of the appeals, to some people at least, of the self-organizing or complex-adaptive-system construal (CAS) is its prospect of providing a naturalistically palatable and nonintentional but nevertheless specific way of discriminating people-cum-computers (and perhaps higher animals) from arbitrary physical devices.

20. In computer science, to take a salient example, the term "the semantics of *X*," where *X* is an expression or construct in a programming language, means approximately the following: the topological (as opposed to geometrical) temporal profile of the behavior to which execution of this program fragment gives rise. By "topological" I mean that the overall temporal order of events is dictated, but that their absolute or metric time-structure (e.g., exactly how fast the program runs) is not. As a result, a program can usually be sped up, either by adjusting the code or running it on a faster processor, without, as is said, "changing the semantics."

21. Best known are Dretske's semantic theory of information (1981), which has more generally given rise to what is known as "indicator semantics"; Fodor's "asymmetrical-dependence" theory (1987); and Millikan's "teleosemantics" or "biosemantics" (1984, 1989). For comparison among these alternatives see, e.g., Fodor (1984) and Millikan (1990).

22. Because formal symbol manipulation is usually defined as "manipulation of symbols independent of their interpretation," some people believe that the formal symbol manipulation construal of computation does not rest on a theory of semantics. But that is simply an elementary, though apparently common, conceptual mistake. As discussed further in section 5, the "independence of semantics" postulated as essential to the formal symbol construal is independence at the level of the phenomenon; it is a claim about how symbol manipulation *works*. Or so at least I believe, based on many years of investigating what practitioners are actually committed to (whether it is *true*—i.e., holds of computation-in-the-wild—is a separate issue). The intuition is simple enough: that semantic properties, such as referring to the Sphinx, or being true, are not of the right sort to do effective work. So they cannot be the sort of property in virtue of the manifestation of which computers *run*. At issue in the present discussion, in contrast, is a more logical form of independence, *at the level of the theory* (or, perhaps, to put it more ontologically and less epistemically, independence at the level of the *types*). Here the formal symbol manipulation construal is as dependent on semantics as it is possible to be: *it is defined in terms of it*. And (as the parent of any teenager knows) defining yourself in opposition to something is not ultimately a successful way of achieving independence. Symbols must have a semantics, in other words (have an actual interpretation, or be interpretable, or whatever), in

order for there to be something substantive for their formal manipulation to proceed independently of. Without a semantic character to be kept crucially in the wings, the formal symbol manipulation construal would collapse in vacuity—would degenerate into something like "the manipulation of structure" or, as I put it in *AOS,* "stuff manipulation"—i.e., materialism.

23. As suggested in the preceding note, philosophers are less likely than computer scientists to expect a theory of computation to be, or to supply, a theory of intentionality. That is, they would not expect the metatheoretic structure to be as expected by most computer scientists and artificial intelligence researchers—namely, to have a theory of intentionality *rest on* a theory of computation. But that does not mean they would necessarily agree with the opposite, which I am arguing here: that a theory of computation will have to rest on a theory of intentionality. Many philosophers seem to think that a theory of computation can be *independent* of a theory of intentionality. Clearly, I do not believe this is correct.

24. It can be tempting to think of the two readings as corresponding to intensional and extensional readings of the phrase "independent of semantics"—but that isn't strictly correct. See *AOS.*

25. See *AOS,* volume 2.

26. Thus Devitt (1991) restricts the computational thesis to what he calls "thought-thought" (t-t) transactions; for him output (t-o) and input (i-t) transactions count as noncomputational.

27. See the preceding note.

28. This statement must be understood within the context of computer science, cognitive science, and the philosophy of mind. It is telling that the term "transducer" is used completely differently in engineering and biology (its natural home), to signify mechanisms that mediate changes in *medium,* not that cross *either* the inside/outside *or* the symbol/referent boundary.

29. See *AOS,* volume 3.

30. See note 20.

31. The fact that it is not a theory of computing does not entail that it does not *apply* to computers, of course. All it means is that, in that application, it is not a theory of them *as computers.*

32. That the so-called theory of computation fails as a theory of computation because it does not deal with computation's intentionality is a result that should be agreed even by someone (e.g., Searle) who believes that computation's intentionality is inherently derivative. I myself do not believe that computation's intentionality is inherently derivative, as it happens, but even those who think that it is must admit that it is still an intentional phenomenon of some sort. For *derivative* does not mean *fake* or *false.* If "derivatively intentional" is not taken to be a substantive constraint, then we are owed (e.g., by Searle) an account of what *does* characterize computation.

33. At one stage I asked a large number of people what they thought "formal" meant—not just computer scientists, but also mathematicians, physicists, sociolo-

gists, etc. It was clear from the replies that the term has very different connotations in different fields. Some mathematicians and logicians, for example, take it to be pejorative, in contrast to the majority of theoretical computer scientists, for whom it has an almost diametrically opposed positive connotation.

34. On its own, an eggplant cannot exactly be either formal or explicit, at least not in its ordinary culinary role, since in that role it is not a representation at all. In fact the only way to make sense of calling something nonrepresentational explicit is as short-hand for saying that it is explicitly represented (e.g., calling eggplant an explicit ingredient of moussaka as a way of saying that the recipe for moussaka mentions eggplant explicitly).

35. Smith (in press).

36. A theory of organization is essentially applied metaphysics.

37. Social construction not as the label for a metaphysical stance, but in the literal sense that we build them.

3

Narrow versus Wide Mechanism

B. Jack Copeland

Editor's Note

Computationalism is grounded in the mechanistic ideas of Descartes, Hobbes, La Mettrie, and others who proposed explanations of minds analogous to those of the prototypical machines (e.g., mechanical clocks) of their time. In the twentieth century, the mechanistic model of the mind became more focused and concentrated on a particular kind of conceptual machine, the Turing machine. This was for various reasons, but mainly because of a famous thesis put forth by Church and Turing to the effect that no human computer (strictly following rules and using only scratch paper and pencil) can out-compute a Turing machine. The "Church-Turing Thesis" eventually led some to believe that the mind is a machine equivalent to a Turing machine, a view Copeland calls "narrow mechanism." This is in contrast to "wide mechanism," the view that the mind is a machine, but possibly a machine that cannot be mimicked by a (universal) Turing machine. Having introduced this distinction, Copeland argues that mechanism per se does not entail narrow mechanism. He also suggests—quoting from various original texts—that Turing himself was not a narrow mechanist. Yet, Turing's work and views on mind have been widely misinterpreted, especially by researchers from within cognitive science. For example, it is not uncommon to find the claim that all functions generated by machines are Turing-machine-computable—called the "Maximality Thesis"—attributed to Turing. Turing, however, did not endorse such a view. He himself defined special machines, so-called oracle-machines that can compute functions that no Turing machine can compute (e.g., the famous "halting function"). Copeland shows that the conflation of evidence for the "Church-Turing Thesis" with evidence for the Maximality Thesis, which he calls the "equivalence fallacy," is widespread in the literature. Furthermore, another related fallacy, which he calls the "Church-Turing fallacy," is common. It is committed by someone who believes that either some result directly established by Church or Turing or the Church-Turing Thesis implies that if mechanism is true, the functions generated by Turing machines provide sufficient mathematical resources for a complete account of human cognition. In particular, many authors seem to believe the following "Thesis S": that any process that can be given a mathematical description can be simulated by a Turing machine. From Newell's physical symbol system hypothesis, to Searle's Chinese room, to Block's

homunculus-head argument, the possibility of wide mechanism (e.g., founded on Turing's oracle machines) seems to be neglected in favor of a narrow mechanist view. As Copeland points out, the question "whether our cognitive architecture, abstracted from resource constraints, is best understood as being a generator of (one or more) Turing-machine-uncomputable functions" remains even if, given the physical restrictions of its implementation, the mind could be simulated by a Turing machine.

1 Historical Mechanism

When Descartes claims that the human body is a machine, he is proposing a novel way of explaining or modeling the functioning of the body. Among the functions that are to be explained in this new way he includes the operation of the sense organs, some amount of internal processing of the resulting sensory ideas, the "stamping of these ideas in the memory," and "the internal movements of the appetites and passions" (*L'Homme:* 108).[1] "I should like you to consider that these functions follow from the mere arrangement of the . . . organs every bit as naturally as the movements of a clock or other automaton follow from the arrangement of its counter-weights and wheels."

Clocks and other mechanical artifacts—such as church organs, water fountains, hydraulically powered statues, and clockwork models of living creatures—were suggesting a new paradigm of explanation in human and animal physiology, in which the functions and (nonvoluntary) movements of the body are viewed as arising "just as [they] would be produced in a machine" (*Fourth Set of Replies:* 161). Even the difference between a living and a dead body is to be understood in terms of "the difference between, on the one hand, a watch or other automaton . . . when it is wound up . . . and, on the other hand, the same watch or machine when it is broken and the principle of its movement ceases to be active" (*Passions of the Soul:* 329–330).

Hobbes's mechanist philosophy amounts in its essentials to a materialist view of mind coupled with the view that the artifactual mode of explanation—typified by the explanation of the working of a clock in terms of the nature and arrangement of its internal parts—may usefully be transferred to the study of naturally occurring systems, including our bodies and our minds: "[W]hat is the Heart, but a Spring; and the Nerves, but so many Strings; and the Joynts, but so many Wheeles, giving

motion to the whole Body, such as was intended by the Artificer?" (*Leviathan:* 1).

In 1748, La Mettrie (*L'Homme machine;* tr. in Thomson 1996) likewise propounded a thoroughgoing mechanism, insisting against Descartes that human beings, and not just their bodies, are machines: "Let us then conclude boldly that man is a machine and that there is in the whole universe only one diversely modified substance" (Thomson 1996: 39). La Mettrie gleefully applied the artifactual mode of explanation to both body and mind:

> [T]he human body is a clock. . . . [T]he heart . . . is like the mainspring of the machine. (ibid.: 34)
>
> [S]ince all the soul's faculties depend so much on the specific organisation of the brain and of the whole body that they are clearly nothing but that very organisation, the machine is perfectly explained! . . . Some wheels, a few springs more than in the most perfect animals, the brain proportionately closer to the heart and thus receiving more blood. . . . (ibid.: 26)

The core of the claim, as put forward by the historical mechanists, that such-and-such naturally occurring item is a machine is this: the item's operation can be accounted for in monistic, materialist terms and in a manner analogous to that in which the operation of an artifact is explained in terms of the nature and arrangement of its parts.[2] I shall refer to the proposition that the mind is a machine in the core sense of "machine" embraced by the historical mechanists as *historical mechanism*.[3]

2 Twentieth-Century Mechanism

Under the influence of work by Alan Turing and, to a lesser extent, Alonzo Church, mechanism took something of a wrong turn in the twentieth century, or, at any rate, was steered for no very good reason in one particular direction to the exclusion of others. In 1936, Turing published his account of what he (later) called "logical computing machines," which Church, in a review, dubbed "Turing machines."[4] A Turing machine consists of a potentially infinite paper tape on which is written a finite number of discrete (e.g., binary) symbols and a scanner that moves back and forth along the tape symbol by symbol reading what if finds and writing further symbols. Turing proved that one particular Turing machine, which he referred to as a "universal computing machine," can

be programmed to perform any task that any other Turing machine can perform. Twelve years later, thanks to developments in high-speed automatic switching, the universal computing machine became a reality.

The effect of Turing's invention on mechanist thinking was marked. The classic paper of J. R. Lucas (1961) affords an excellent window on the state of affairs that existed twenty-five years after Turing's invention. Lucas wished to refute mechanism: "Mechanism is false, that is, . . . minds cannot be explained as machines" (112). To do so, it was, he thought, sufficient to argue that the powers of the mind exceed those of a universal Turing machine, for "any system which [is] not a Turing machine [is] not a machine within the meaning of the act" (126).[5] (Strictly, of course, Lucas should have said that any system that cannot be *simulated,* or exactly mimicked, by a Turing machine is not a machine within the meaning of the act, for this was surely his intention; but the practice of suppressing the word "simulated" and its cognates in such a context as this is a common one, and will sometimes be followed here.) Within twenty-five years of Turing's invention, the idea that the mind is a Turing machine had become central to the mechanist conception of mind, so much so that one of mechanism's foremost critics needed to take no cognizance of alternative mechanist conceptions. Indeed, by the time Lucas was writing, it had become endemic to mechanist thinking that there could be no alternative mechanist conception. The absence of any alternative mechanist conception was, and still is, held to be entailed by a principle supported by the logical discoveries of Turing and Church. A typical formulation of this supposed principle is: the class of possible operations that can be carried out by information-processing machinery is identical to the class of operations that can be carried out by a universal Turing machine. Turing's biographer, Hodges, gives expression to the common perception of matters when he writes: "Alan had . . . discovered something almost . . . miraculous, the idea of a universal machine that could take over the work of *any* machine. . . . So there could be a single machine which, by reading the descriptions of other machines placed upon its 'tape,' could perform the equivalent of human mental activity" (1992: 109).

The view that the universal Turing machine is in some appropriate sense *maximal* among machines is widespread in the philosophical, cognitive, and biological literature. For example, the following version of the

view is from a manifesto of the Artificial Life movement (Langton 1989: 12). "There are certain behaviors that are 'uncomputable'—behaviors for which *no* formal specification can be given for a machine that will exhibit that behavior. The classic example of this sort of limitation is Turing's famous *Halting Problem:* can we give a formal specification for a machine which, when provided with the description of *any* other [*sic*] machine together with its initial state, will . . . determine whether or not that machine will reach its halt state? Turing proved that no such machine can be specified." As we shall see, it is far from the case that Turing proved any such thing.

Those who have equated machines "within the meaning of the act" with Turing machines would no doubt resist any account of themselves as revisionists who have replaced historical mechanism with a narrower thesis of their own devising. They may claim that they have merely been explicit about a point to which historical mechanism was always committed, unbeknownst to its sixteenth- and seventeenth-century supporters: had La Mettrie, for example, known of the logical discoveries of Turing and Church, he would have accepted that his mechanism entails that the mind is a Turing machine. I shall argue that the view that modern Turing-machine mechanism is simply a clarified version of historical mechanism is in error. One can uphold historical mechanism and deny Turing-machine mechanism without contradicting oneself. Let me coin the term *narrow mechanism* for the view that the mind is (strictly, can be simulated by) a Turing machine. A *wide mechanist,* on the other hand, holds that the mind is a machine but countenances the possibility of information-processing machines that cannot be mimicked by a universal Turing machine, and allows in particular that the mind may be such a machine. Elsewhere, I term such machines *hypercomputers* (Copeland and Proudfoot 1999). I shall argue that the widespread acceptance of narrow mechanism among mechanists represents an unwarranted circumscription of the mechanist tradition. The view that this circumscription is somehow necessitated by the work of Church and Turing is a muddle. Specifically, I shall be claiming that: (1) mechanism does not entail narrow mechanism; (2) Turing himself, a mechanist par excellence, was not a narrow mechanist; (3) neither the Church-Turing Thesis, nor any other formal or semiformal result of Church or Turing, favors narrow over

wide mechanism; (4) typical arguments for narrow mechanism are vitiated by what I have called elsewhere the *Church-Turing fallacy* (Copeland 1998d).

Lately there have been encouraging signs that the grip of narrow mechanism is loosening. The newly emerging field known as UMC (unconventional models of computation) explores computational approaches to cognition that transgress the boundaries of narrow mechanism (see Calude, Casti, and Dinneen 1998).[6] Also the recent *Dynamical Hypothesis* in cognitive science (van Gelder 1995, 1998) is a wide mechanist hypothesis. The new dynamicists distance themselves from the mainstream computational approach by pointing out that the *Dynamical Hypothesis* countenances dynamical (and, specifically, cognitive) systems whose behavior cannot—even in principle—be calculated by a Turing machine (van Gelder 1998, section 6.3).

3 Computers and Computers

It has often been remarked that when Turing uses the word "computer" in his early papers, he does not employ it in its modern sense. Many passages make this obvious, for example the following: "Computers always spend just as long in writing numbers down and deciding what to do next as they do in actual multiplications, and it is just the same with ACE. . . . [T]he ACE will do the work of about 10,000 computers. . . . Computers will still be employed on small calculations" (Turing 1947: 116, 120). (The ACE or Automatic Computing Engine was an electronic stored-program computer designed by Turing and built at the National Physical Laboratory, London. A pilot version first ran in 1950 and at the time was the fastest computer in the world.) Turing introduces his "logical computing machines" with the intention of providing an idealized description of a certain human activity, the tedious one of *numerical computation,* which until the advent of automatic computing machines was the occupation of many thousands of people in commerce, government, and research establishments. These people were referred to as *computers*. Turing prefaces his first description of a Turing machine with the words: "We may compare a man in the process of computing a . . . number to a machine" (1936: 231).

The Turing machine is a model, idealized in certain respects, of a human computer. Wittgenstein put this point in a striking way: "Turing's 'Machines.' These machines are *humans* who calculate" (Wittgenstein 1980, §1096). It is a point that Turing was to emphasize, in various forms, again and again. For example: "A man provided with paper, pencil, and rubber, and subject to strict discipline, is in effect a universal machine" (Turing 1948: 9). The electronic stored-program digital computers for which the universal Turing machine was a blueprint are, each of them, computationally equivalent to a Turing machine with a finite tape, and so they too are, in a sense, models of human beings engaged in computation. Turing chose to emphasize this when explaining the new electronic machines in a manner suitable for an audience of uninitiates: "The idea behind digital computers may be explained by saying that these machines are intended to carry out any operations which could be done by a human computer" (1950a: 436). He makes the point a little more precisely in the technical document containing his preliminary design for the ACE: "The class of problems capable of solution by the machine can be defined fairly specifically. They are [a subset of] those problems which can be solved by human clerical labour, working to fixed rules, and without understanding" (Turing 1945: 38–39). (Turing went on to characterize the subset in terms of the amount of paper and time available to the human clerk.) It was presumably because he considered the point under discussion to be essential for understanding the nature of the new electronic machines that he chose to begin his *Programmers' Handbook for Manchester Electronic Computer* with this explanation: "Electronic computers are intended to carry out any definite rule of thumb process which could have been done by a human operator working in a disciplined but unintelligent manner" (Turing 1950b: 1).

It was not some deficiency of imagination that led Turing to model his logical computing machines on what could be achieved by a human computer. The purpose for which the Turing machine was invented demanded it. Turing introduced the Turing machine in the course of arguing that the *Entscheidungsproblem,* or decision problem, for the predicate calculus—posed by Hilbert (Hilbert and Ackermann 1928)—is unsolvable. Here is Church's account of the *Entscheidungsproblem:* "By the Entscheidungsproblem of a system of symbolic logic is here understood

the problem to find an effective method by which, given any expression Q in the notation of the system, it can be determined whether or not Q is provable in the system" (Church 1936a: 41).

"Effective" and its synonym "mechanical" are terms of art in mathematical logic. A mathematical method is termed "effective" or "mechanical" if and only if it can be set out in the form of a list of instructions able to be followed by an obedient human clerk—the computer—who works with paper and pencil, reliably but without insight or ingenuity, for as long as necessary. The truth table test is such a method for the propositional calculus. Turing showed by means of a two-stage argument that there can be no such method in the case of the predicate calculus. First, he proved formally that there is no Turing machine that can determine, in a finite number of steps, whether or not any given formula *Q* of the predicate calculus is a theorem of the predicate calculus. Second, he argued *informally* for the proposition that whenever there is an effective method for performing a mathematical task, then the method can be carried out by a Turing machine in some finite number of steps. These two stages jointly secure the result that there is no effective method for determining whether or not an arbitrary formula *Q* of the predicate calculus is a theorem of the calculus.

Notice that this result does not entail that there can be no *machine* for determining this (contrary to various writers). The *Entscheidungsproblem* for the predicate calculus is the problem of finding a humanly executable procedure of a certain sort, and the fact that there is none is entirely consistent with the claim that some machine may nevertheless be able to decide arbitrary formulae of the calculus; all that follows is that such a machine, if it exists, cannot be mimicked by a human computer. Turing's (and Church's) discovery was that there are limits to what a *human computer* can achieve; for all that, their result is often portrayed as a discovery concerning the limitations of mechanisms in general.

The proposition that any effective method can be carried out by a Turing machine is known variously as *Turing's Thesis* and the *Church-Turing Thesis*. Turing stated his thesis in numerous places, with varying degrees of rigor. The following formulation is one of the most accessible:

> LCMs [logical computing machines] can do anything that could be described as "rule of thumb" or "purely mechanical." (Turing 1948: 7)

Turing adds "This is sufficiently well established that it is now agreed amongst logicians that 'calculable by means of an LCM' is the correct accurate rendering of such phrases" (ibid.).

Church proposed the (not quite) equivalent thesis that whenever there is an effective method for calculating the values of a function on the positive integers, then the function is recursive (not quite equivalent because Turing did not restrict attention to functions on the positive integers, mentioning also "computable functions of a real or computable variable, computable predicates, and so forth") (Church 1936b: 356; Turing 1936: 230). The term "Church-Turing Thesis" seems to have been introduced by Kleene (with a small flourish of bias in favor of his mentor Church): "So Turing's and Church's theses are equivalent. We shall usually refer to them both as *Church's thesis,* or in connection with that one of its . . . versions which deals with 'Turing machines' as the *Church-Turing thesis*" (Kleene 1967: 232).

Essentially, then, the Church-Turing thesis says that no human computer, or machine that mimics a human computer, can out-compute a universal Turing machine. A further proposition, very different from this—namely that a Turing machine can compute whatever can be computed *by any machine*—is nowadays sometimes referred to as the Church-Turing Thesis or as Church's Thesis. For example, Smolensky says: "connectionist models . . . may possibly even challenge the strong construal of Church's Thesis as the claim that the class of well-defined computations is exhausted by those of Turing machines" (Smolensky 1988: 3). This loosening of established terminology is unfortunate, for neither Church nor Turing endorsed, or even formulated, this further proposition. There are numerous examples of this and other extended usages in the literature. The following are typical.

> That there exists a most general formulation of machine and that it leads to a unique set of input-output functions has come to be called *Church's thesis.* (Newell 1980: 150)

> Church-Turing thesis: If there is a well defined procedure for manipulating symbols, then a Turing machine can be designed to do the procedure. (Henry 1993: 149)

> [I]t is difficult to see how any language that could actually be run on a physical computer could do more than Fortran can do. The idea that there is no such language is called Church's thesis. (Geroch and Hartle 1986: 539)

A typical way of stating Church's thesis is the following: In an ideal world the limit of computation is exactly captured by Turing computability. (Hogarth 1994: 133)

More distant still from anything that Church or Turing actually wrote:

The first aspect that we examine of Church's Thesis . . . [w]e can formulate, more precisely: The behaviour of any discrete physical system evolving according to local mechanical laws is recursive. (Odifreddi 1989: 107)

I can now state the physical version of the Church-Turing principle: Every finitely realizable physical system can be perfectly simulated by [Turing's] universal model computing machine. . . . This formulation is both better defined and more physical than Turing's own way of expressing it. (Deutsch 1985: 99)

It is important to distinguish between Turing's thesis and the stronger proposition that whatever functions (in the mathematical sense of "function") can be generated by machines can be generated by a universal Turing machine.[7] (To say that a function f can be generated by a machine m is simply to say that for each of the function's arguments, x, if x is presented to m as input, m will carry out some finite number of atomic processing steps at the end of which it produces the corresponding value of the function, $f(x)$.) I shall call this stronger proposition the "Maximality Thesis" ("Thesis M") and shall use expressions such as "the Church-Turing Thesis properly so called" for the proposition that Church and Turing themselves endorsed.[8]

Maximality Thesis: All functions that can be generated by machines (working on finite input in accordance with a finite program of instructions) are Turing-machine-computable.

Thesis M itself admits of two interpretations, according to whether the phrase "can be generated by a machine" is taken in the this-worldly sense of "can be generated by a machine that conforms to the physical laws (if not to the resource constraints) of the actual world," or in a sense that abstracts from whether or not the notional machine in question could exist in the actual world. The former version of thesis M is an empirical proposition whose truth-value is unknown. The latter version of thesis M is known to be false. As I explain in the next section, there are notional machines that generate functions that no Turing machine can generate.

As previously remarked, the word "mechanical," in technical usage, is tied to effectiveness, "mechanical" and "effective" being used inter-

changeably. (Gandy 1988 has outlined the history of this usage of the word "mechanical.") Thus, statements like the following are to be found in the technical literature:

> Turing proposed that a certain class of abstract machines could perform any "mechanical" computing procedure. (Mendelson 1964: 229)

Understood correctly, this remark attributes to Turing not thesis M but the Church-Turing Thesis properly so called. This usage of "mechanical" tends to obscure the possibility that there may be machines, or biological organs, that generate (or compute, in a broad sense) functions that cannot be computed by Turing machine. For the question "Can a machine execute a procedure that is not mechanical?" may appear self-answering; yet this is precisely what is asked if thesis M is questioned.[9]

In the technical literature, the word "computable" is sometimes tied by definition to effectiveness: a function is said to be computable if and only if there is an effective procedure for determining its values. The Church-Turing thesis then becomes:

> Every computable function can be computed by Turing machine.

Corollaries such as the following are sometimes offered:

> certain functions are uncomputable in an absolute sense: uncomputable even by [a Turing machine], and, therefore, uncomputable by any past, present, or future real machine. (Boolos and Jeffrey 1980: 55)

Of course, the decision to tie the term "computable" and its cognates to the concept of effectiveness does not settle the truth-value of thesis M; rather, those who abide by this terminological decision are prevented from describing any machine that falsifies thesis M as *computing* the function that it generates. Yet to a casual reader of the technical literature, statements like the one just quoted may appear to say more than they in fact do.

Putnam, himself at one time a narrow mechanist, is one of the few writers on the philosophy of mind to question the maximality thesis:

> materialists are committed to the view that a human being is—at least metaphorically—a machine. It is understandable that the notion of a Turing machine might be seen as just a way of making this materialist idea precise. Understandable, but hardly well thought out. The problem is the following: a "machine" in the sense of a physical system obeying the laws of Newtonian physics need not be a Turing machine. (Putnam 1992: 4)

4 Turing's Other Machines

In his doctoral thesis (which was supervised by Church), Turing introduced the idea of machines able to solve mathematical problems that cannot be solved by the "logical computing machines" of his 1936 paper (1939, sect. 4). He described these as "a new kind of machine" and called them "O-machines" (1939: 173).

An O-machine is in essence an ordinary Turing machine augmented with a black box that generates some function that cannot be generated by a Turing machine.[10] Turing refers to the black box as an "oracle" (ibid.). As in the case of an ordinary Turing machine, the behavior of an O-machine is determined by a table of instructions (or program). The table provides an exhaustive specification of which fundamental processes the machine is to perform when it is in such-and-such state and has such-and-such symbol in its scanner. The tables of the two sorts of machine differ only in the following respect: an O-machine table may contain instructions of the form "*TRANSFORM* #*". "#*" refers to some particular string of symbols on the machine's tape, the beginning of the string being marked by the presence on the tape of a reserved symbol "#" and the end of the string being marked by a reserved symbol "*". The instruction causes the portion of tape so marked to be presented to the black box. The symbols on this portion of tape constitute a specification of an argument of whatever function it is that the box generates (or of a series of *n* arguments in case the function is *n*-ary). The box replaces the symbols with a specification of the corresponding value of the function.

One way of conceptualizing an oracle—which need not reflect the box's actual manner of functioning—is as a device accessing an infinite internal tape on which there have been inscribed, in order, all the infinitely many arguments and values of whatever function it is that the oracle generates. This device can produce any of the function's values after only a finite search along the tape.

The transform operation performed by the oracle is, in Turing's expression, one of the "fundamental processes" of the machine (1939: 173).[11] He gave no indication of how this process might conceivably be carried out, saying only that an oracle works by "unspecified means"

and that "we shall not go any further into the nature of [an] oracle" (ibid., pp. 172–173). In fact, notional machinery that discharges the task of an O-machine's black box is not hard to concoct. Suppose for the sake of illustration that the function generated by the box is the Halting function. The function is easily explained. Assume the Turing machines to be ordered in some way, so that we may speak of the first Turing machine in the ordering, the second, and so on (there are various standard ways of accomplishing this ordering). The arguments of the Halting function are simply 1, 2, 3, ... The value of the function for any argument n is 1 if and only if the nth Turing machine eventually comes to the end of its computation and halts, and is 0 if and only if the nth machine runs on forever (as would a Turing machine programmed to produce in succession the digits of the decimal representation of π, for instance). No Turing machine can generate the Halting function.

It is convenient to write "h_n" to represent the value of the Halting function for argument n. h_n is always 0 or 1. Consider the following decimal specification of a number: $0{\cdot}h_1h_2h_3\ldots$; I call this number "τ" (for Turing).[12] (The first few digits of τ might be $0{\cdot}000000011\ldots$) Like π, τ is a definite—irrational—number.[13] The magnitude of some physical quantity might conceivably be exactly τ units. Suppose that some mechanism A does store exactly τ units of such a physical quantity, which for the sake of vividness one might call "charge." Suppose further that a mechanism B can measure the quantity of "charge" stored in A to any specified number of significant figures. A and B jointly discharge the task of an oracle that generates the Halting function. B determines h_n by measuring A's charge to an appropriate number of significant figures and outputting the nth digit of the result.

An O-machine consisting of a Turing machine and the oracle just described is a machine in the sense that its behavior is the product of the nature and arrangements of its material parts. The core claim of historical mechanism—namely, that the mind is wholly explicable within the resources of some monistic, materialist theory in a manner analogous to that in which the behavior of artifacts is accounted for in terms of the organization and functions of their parts—is evidently consistent with the hypothesis that the mind is an O-machine.

5 Arguments That Narrow Mechanism Exhausts Mechanism

I have elsewhere coined the term *Church-Turing fallacy* for a persistent error that is to be encountered in modern writing concerning mechanism (1998d). This fallacy takes a number of distinct but closely related forms, several of which are described below. Far from being confined to writers of one particular philosophical stripe, the Church-Turing fallacy is to be found in the work of thinkers of very different persuasions. I shall illustrate the widespread nature of the fallacy by considering arguments put forward by a diverse field of modern thinkers (including Paul and Patricia Churchland, Dennett, Dreyfus, Fodor, Newell, and Searle). It is important that the fallacy be exposed and eradicated. Anyone in its grip will think that narrow mechanism exhausts mechanism: to them, conceptual space will seem to contain no room for mechanical models of the mind that are not equivalent, in the appropriate sense, to one or another class of Turing machines. Propagation of the fallacy by leading theoreticians has assisted in blotting from view a potentially rich field of possible models of human cognition.

In essence, to commit the Church-Turing fallacy is to believe that the Church-Turing thesis, or some formal or semiformal result established by Turing or Church, secures the proposition that, if mechanism is true, the functions generated by Turing machines provide sufficient mathematical resources for a complete account of human cognition. One form of the fallacy concerns specifically the notion of a Turing machine simulating, or mimicking, the human cognitive apparatus.[14] Someone commits this form of the fallacy (the *simulation fallacy*) by believing that the Church-Turing Thesis—or, again, some formal or semiformal result established by Turing or Church—entails that, if mechanism is true, then a universal Turing machine can simulate the mind. The *equivalence fallacy* involves mistaking the strong evidence for the Church-Turing Thesis properly so called for evidence supporting thesis M, and so passing from various logico-mathematical considerations to the view that narrow mechanism exhausts mechanism.

The Church-Turing fallacy has led to some remarkable claims in the foundations of psychology. For example, one frequently encounters the view that psychology must be capable of being expressed in computational terms, and so ultimately in terms of the computational properties

of Turing machines. Yet it is certainly possible that psychology will find need to employ mathematical functions that cannot be generated by Turing machine.

Fodor writes:

> Although the elementary operations of the Turing machine are restricted, iterations of the operations enable the machine to carry out any well-defined computation on discrete symbols. . . . If a mental process can be functionally defined as an operation on symbols, there is a Turing machine capable of carrying out the computation . . . The "black boxes" that are common in flow charts drawn by psychologists often serve to indicate postulated mental processes for which Turing reductions are wanting. Even so, the possibility in principle of such reductions serves as a methodological constraint on psychological theorizing by determining what functional definitions are to be allowed. (1981: 130; see also 1983: 38–39)

The claim made in the second sentence of this quotation is false.[15] Each O-machine carries out some well-defined operation on discrete symbols. As in the case of an ordinary Turing machine, an O-machine generates discrete symbolic output from discrete symbolic input, possibly via intermediate structures of discrete symbols, by means of the step-by-step procedure specified in its machine table. Fodor's overarching view that mental processes consist of operations on discrete symbols does not entail the narrow mechanist view of psychology which he advocates.

The Churchlands hold that Turing's "results entail something remarkable, namely that a standard digital computer, given only the right program, a large enough memory and sufficient time, can compute *any* rule-governed input-output function. That is, it can display any systematic pattern of responses to the environment whatsoever" (Churchland and Churchland 1990: 26). If this were true then the view that psychology must be capable of being expressed in standard computational terms would be secure. But Turing had no result entailing this. What he did have was a result entailing the exact opposite. The theorem that no Turing machine can generate the Halting function entails that there are possible patterns of responses to the environment, perfectly systematic patterns, which no Turing machine can display. The Halting function is a mathematical characterization of just such a pattern.

Searle argues for a narrow mechanist account of mind directly from Church's thesis:

> Can the operations of the brain be simulated on a digital computer? . . . [G]iven Church's Thesis that anything that can be given a precise enough characterization

as a set of steps can be simulated on a digital computer, it follows trivially that the question has an affirmative answer. (1992: 200)

If the question ["Is consciousness computable?"] asks "Is there some level of description at which conscious processes and their correlated brain processes can be simulated [by a Turing machine]?" the answer is trivially yes. Anything that can be described as a precise series of steps can be simulated [by a Turing machine]. (1997: 87)

Of course, Church's Thesis properly so called does *not* say that anything that can be described as a precise series of steps can be simulated by Turing machine. The behavior of an O-machine as it follows the instructions in its table can certainly be characterized as consisting of a set of steps, each step consisting of the execution of one of the machine's fundamental processes; yet if the brain is an O-machine, it is false that its operations can (in their entirety) be simulated by Turing machine.

As I have already said, the error that Searle commits here—of holding that Church's and Turing's results somehow entail that the brain can be *simulated* by a Turing machine—is a common one. The entry on Turing in Guttenplan's *A Companion to the Philosophy of Mind* contains the following claims: "we can depend on there being a Turing machine that captures the functional relations of the brain," for so long as "these relations between input and output are functionally well-behaved enough to be describable by . . . mathematical relationships . . . we know that some specific version of a Turing machine will be able to mimic them" (Guttenplan 1994: 595). Even Dreyfus, in the course of *criticizing* the view that "man is a Turing machine," succumbs to the belief that it is a "fundamental truth that every form of 'information processing' (even those which *in practice* can only be carried out on an 'analogue computer') must *in principle* be simulable on a [Turing machine]" (1992: 195). Similarly, Johnson-Laird and the Churchlands argue:

If you assume that [consciousness] is scientifically explicable . . . [and] [g]ranted that the [Church-Turing] thesis is correct, then the final dichotomy rests on Craik's functionalism. If you believe [functionalism] to be false . . . then presumably you hold that consciousness could be modelled in a computer program in the same way that, say, the weather can be modelled. . . . If you accept functionalism, however, then you should believe that consciousness is a computational process. (Johnson-Laird 1987: 252)

Church's Thesis says that whatever is computable is Turing computable. Assuming, with some safety, that what the mind-brain does is computable, then it can in principle be simulated by a computer. (Churchland and Churchland 1983: 6)

As previously mentioned, Churchland and Churchland believe, erroneously, that Turing's "results entail . . . that a standard digital computer . . . can . . . display any systematic pattern of responses to the environment whatsoever" (1990: 26). This no doubt explains why they think they can assume "with some safety" that what the mind-brain does is computable, for on their understanding of matters this is to assume only that the mind-brain is characterized by a "rule-governed" (ibid.) input/output function.

Each of the authors quoted appears to be assuming the truth of a close cousin of thesis M, which I shall call

Thesis S: Any process that can be given a mathematical description (or a "precise enough characterization as a set of steps," or that is scientifically describable or scientifically explicable) can be simulated by a Turing machine.

As with thesis M, neither the Church-Turing Thesis properly so called nor any result of Turing or Church entails thesis S. This is so even when thesis S is taken as concerning only processes that conform to the physics of the real world. Taken in a broader sense that abstracts from the issue of whether or not the processes in question could exist in the actual world, thesis S is known to make a false claim (the processing carried out by an O-machine suffices as a counterexample to it). The view that the mind is scientifically explicable in no way entails narrow mechanism. For all we currently know, a completed neuroscience may present the mind-brain as a machine that generates functions that no Turing machine can generate.

Paramount among the evidence for the Church-Turing Thesis properly so called is the fact that all attempts to give an exact analysis of the intuitive notion of an effective mathematical method have turned out to be equivalent in extension. Because of the prima facie diversity of the various analyses, their equivalence is generally considered extremely strong evidence for the Church-Turing Thesis properly so called (see, for example, Kleene 1952, chs. 12, 13). (Apart from Turing's analysis, and Church's analyses in terms of lambda-definability and recursiveness [Church 1936b], there are analyses in terms of register machines [Shepherdson and Sturgis 1963], Post's canonical and normal systems [Post 1943, 1946], combinatory definability [Schönfinkel 1924; Curry 1929, 1930, 1932], Markov algorithms [Markov 1960], and Gödel's notion of reckonability

[Gödel 1936; Kleene 1952].) In the narrow mechanist literature, the equivalence of these diverse analyses is commonly taken to be evidence for thesis M. This is nothing more than a confusion—the equivalence fallacy. The analyses under discussion are of the notion of an effective method, not of the notion of a machine-generable function; the equivalence of the analyses bears only on the issue of the extent of the former notion and indicates nothing concerning the extent of the latter.

Newell and Simon (1976) encapsulate a narrow mechanist view of mind in their famous *physical symbol system hypothesis:*

> A physical symbol system has the necessary and sufficient means for general intelligent action. . . . [A]ny physical symbol system of sufficient size can be organized further to exhibit general intelligence. (116)

By the phrases "general intelligent action" and "general intelligence" Newell and Simon "wish to indicate the same scope of intelligence as we see in human action" (ibid.). A physical symbol system is a universal Turing machine, or any equivalent system, situated in the physical (as opposed to the conceptual) world. (The tape of the machine is accordingly finite; Newell specifies that the storage capacity of the tape, or equivalent, be unlimited in the practical sense of finite yet not small enough to "force concern" [Newell 1980: 161; see also Turing 1948: 15].) The physical symbol system hypothesis is the foundation stone of the particular brand of narrow mechanism that has been the dominant research paradigm in cognitive science since mid–twentieth century.

Newell thinks it is easily established that a physical symbol system can be organized to exhibit general intelligence: "A [physical symbol] system always contains the potential for being any other system if so instructed. Thus, a [physical symbol] system can become a generally intelligent system" (1980: 170).

Is the premise true? A physical symbol system, being a *universal* Turing machine situated in the real world, can, if suitably instructed, simulate (or, metaphorically, become) any other physical symbol system (modulo some fine print concerning storage capacity). If this is what the premise means, then it is true; but if taken literally, the premise is false, for systems can be specified which no physical symbol system can simulate (for example, an O-machine). If the premise is interpreted in the former manner, however, the argument is simply a non sequitur. Only to one who be-

lieves, as Newell does, that "the notion of machine or determinate physical mechanism" is "formalized" by the notion of a Turing machine (1980: 170) will the argument appear deductively valid. His defense of his view that the universal Turing machine exhausts the possibilities of mechanism involves an example of the equivalence fallacy:

> [An] important chapter in the theory of computing . . . has shown that all attempts to . . . formulate . . . general notions of mechanism . . . lead to classes of machines that are equivalent in that they encompass in toto exactly the same set of input-output functions. In effect, there is a single large frog pond of functions no matter what species of frogs (types of machines) is used. . . . A large zoo of different formulations of maximal classes of machines is known by now—Turing machines, recursive functions, Post canonical systems, Markov algorithms. . . . (1980: 150)

Dennett (1978: 83) has fielded the following argument from "Church's Thesis," which he states in the form "anything computable is Turing-machine computable":

> [A] non-question-begging psychology will be a psychology that makes no ultimate appeals to unexplained intelligence, and that condition can be reformulated as the condition that whatever functional parts a psychology breaks its subjects into, the smallest, or most fundamental, or least sophisticated parts must not be supposed to perform tasks or follow procedures requiring intelligence. That condition in turn is surely strong enough to ensure that any procedure admissible as an "ultimate" procedure in a psychological theory falls well within the intuitive boundaries of the "computable" or "effective" as these terms are . . . used in Church's Thesis. . . . [A]ny psychology that stipulated atomic tasks that were "too difficult" to fall under Church's Thesis would be a theory with undischarged homunculi.

The conclusion Dennett draws from the argument is that "the supposition that there might be a non-question-begging non-mechanistic psychology gets you nothing" (ibid.: 83, and see also 112); and, clearly, if the argument worked, it would also show that there cannot be a non-question-begging mechanistic psychology postulating atomic processes that are not Turing-machine computable. The *transform* operation discussed in section 4 serves to highlight the error in the argument: this operation is an example of an atomic task "too difficult to fall under Church's thesis," yet the account of its implementation in terms of mechanisms *A* and *B* is entirely mechanical and makes no "appeal to unexplained intelligence." At bottom, what has led Dennett astray is his belief that Church's thesis tells us that every "task for which there is a clear recipe composed of

simple steps can be performed by a very simple computer, a universal Turing machine, the universal recipe-follower" (ibid.: xviii).

It is worth remarking that a number of well-known arguments against computational functionalism cannot be brought to bear when it is wide rather than narrow mechanism that informs the functionalist's account. Examples are Searle's Chinese room argument (discussed in Copeland 1998d) and Block's homunculus-head argument. Block too eagerly offers the latter as an embarrassment "for all versions of functionalism" (1978: 277). His brain-of-slaves scenario is well known: a billion Chinese clerks working effectively are brought into functional equivalence with your mind, each clerk implementing a single line "of an adequate machine table that describes you" (1978: 278). Here Block tacitly assumes narrow mechanism. His argument is powerless against the richer functionalism countenanced by wide mechanists, which allows that the machine table describing your mind may be of such a nature that—like the table of an O-machine—it cannot be implemented by human clerks working effectively (see section 3).

6 Some Potentially Misleading Features of Turing's Presentation

Turing more than anyone else is to be thanked for uniting historical mechanism with modern mathematics. He enriched mechanism with an abstract theory of (information-processing) machines, presenting us with an indefinitely ascending hierarchy of possible machines, of which the Turing machines form the lowest level. His work posed a new question: If the mind is a machine, where in the hierarchy does it lie? Yet Turing has been widely misinterpreted. He is popularly believed to have proven some limitative result concerning the extent of the class of possible machines; and, as we have seen, expressions of the view that mechanism entails narrow mechanism are generally accompanied by a nod toward Turing (or Church).

Precisely how these misunderstandings of Turing's work arose is a matter of little consequence. Part of the explanation, perhaps, is the presence of various minor features of Turing's mode of presentation that can easily mislead. One of these has already been mentioned: in Turing's early papers the words "computer," "computable," and "computation" are employed not in their modern sense as pertaining to machines but as

pertaining to human calculators. So, for example, when Turing maintains that every number or function "which would naturally be regarded as computable" (1936: 249) can be generated by a Turing machine, he is advancing only the Church-Turing Thesis properly so called, and not a version of thesis M. Similarly, Turing's use of the phrase "universal computing machine" implies nothing more than that the machine so denoted can carry out the work of any human computer.

When Turing uses the word "machine," he often means not machine-in-general but logical computing machine or, simply, effective method. At one point, he explicitly draws attention to this usage: "The expression 'machine process' of course means one which could be carried out by the type of machine I was considering [in Turing 1936]" (Turing 1947: 107). Thus when, a few pages later, he asserts that "machine processes and rule of thumb processes are synonymous" (112), he is to be understood not as advocating narrow mechanism but as advancing the Church-Turing Thesis properly so called (and its converse). Likewise, when he says that an oracle "cannot be a machine," he probably means only that an oracle cannot be a Turing machine (as he himself had proved); he remarks in the very next sentence that an O-machine is "a new kind of machine" (1939: 173). Especially liable to mislead are statements like the following, which a casual reader might easily mistake for the claim that the universal Turing machine provides a "general notion of mechanism" (Newell's phrase):

> The importance of the universal machine is clear. We do not need to have an infinity of different machines doing different jobs. A single one will suffice. The engineering problem of producing various machines for various jobs is replaced by the office work of "programming" the universal machine to do these jobs. (Turing 1948: 7)

In context, it is clear that these remarks of Turing's concern machines equivalent to logical computing machines.

Turing introduces the term "discrete-state machine" for those machines whose possible states (configurations) form a discrete set: these machines "move by sudden jumps or clicks from one quite definite state to another" (1950a: 439; 1948: 5). (Each Turing machine is a discrete-state machine, of course.) He opposes the discrete-state machines to "continuous machinery," the states of which "form a continuous manifold, and the behavior of the machine is described by a curve on this manifold"

(1948: 5).[16] It is sometimes said that any discrete-state machine can be simulated by a universal Turing machine.[17] Turing himself may appear to be endorsing this claim (which is a restricted form of thesis M). He says: a "digital computer could mimic the behaviour of any discrete-state machine" (1950a: 441). The surrounding discussion (440–441) makes it clear that he intends this statement to apply only in the case of those discrete-state machines that have "a finite number of possible states" (that is, a finite number of possible configurations) (440). He points out that when this condition is satisfied, the behavior of the machine can be described exhaustively by a finite table of the sort nowadays commonly called a "look-up" table (440); it is on the basis of being "[g]iven the table corresponding to a discrete-state machine" that a digital computer could mimic the latter (441).[18]

An example of a discrete-state machine whose behavior cannot be calculated by a universal Turing machine is a digital computer with an infinite-capacity store and what Turing calls "a random element" (1950a: 438–439). He refers to computing machines with a random element as "partially random machines" (1948: 9).

7 Turing's View: The Mind as Partially Random Machine

A device that outputs a genuinely random and unboundedly long sequence of integers is a form of oracle (section 4). As suggested previously, the device may be conceptualized as one accessing a tape on which an infinite random sequence of integers has been inscribed.[19] Turing explains that a discrete-state machine to which such a device is attached may be set up so as to choose between two paths of action by calling to the device for a number and following one path if, say, the number is even and the other if it is odd (1948: 9). Except in the case where the number of possible configurations of the machine is finite, a partially random discrete-state machine cannot be simulated by a Turing machine, for as Church pointed out in 1939, if a sequence of integers $a_1, a_2, \ldots, a_n, \ldots$ is random, then there is no function $f(n) = a_n$ that is calculable by a Turing machine (Church 1940: 134–135).

Turing often mentions this idea of partial randomness. For example, in a paper on machine intelligence he wrote: "[O]ne feature that I would

like to suggest should be incorporated in the machines . . . is a 'random element'. . . . This would result in the behaviour of the machine not being by any means completely determined by the experiences to which it was subjected" (Turing 1951a).[20] Much interested in the issue of freewill, Turing seems to have believed that the mind is a partially random machine. We have the word of one of Turing's closest associates, Max Newman, that Turing "had a deep-seated conviction that the real brain has a 'roulette wheel' somewhere in it."[21] So far as is known, Turing's only surviving discussion of these matters occurs in the typescript of a lecture that he gave in 1951 on BBC radio, entitled "Can Digital Computers Think?" (previously unpublished, the text is now available in my 1998a).[22] In the course of his discussion, Turing considers the claim that if "some particular machine can be described as a brain we have only to programme our digital computer to imitate it and it will also be a brain." He remarks that this "can quite reasonably be challenged," pointing out that there is a difficulty if the behavior of the machine is not "predictable by calculation," and he draws attention to Eddington's view that "no such prediction is even theoretically possible" on account of "the indeterminacy principle in quantum mechanics."

Turing's overarching aim in the lecture is to answer the question posed by his title, and his strategy is to argue for the proposition that "[i]f any machine can appropriately be described as a brain, then any digital computer can be so described." This proposition is consistent, he explains, with the possibility that the brain is the seat of free will:

> To behave like a brain seems to involve free will, but the behaviour of a digital computer, when it has been programmed, is completely determined. . . . [I]t is certain that a machine which is to imitate a brain must appear to behave as if it had free will, and it may well be asked how this is to be achieved. One possibility is to make its behaviour depend on something like a roulette wheel or a supply of radium. . . . It is, however, not really even necessary to do this. It is not difficult to design machines whose behaviour appears quite random to anyone who does not know the details of their construction. (Turing 1951b: 464)

He calls machines of the latter sort "apparently partially random" (1948: 9); an example is a Turing machine in which "the digits of the number τ [are] used to determine the choices" (ibid.).[23] Apparently partially random machines imitate partially random machines. If the brain is a partially random machine, an appropriately programmed digital

computer may nevertheless give a convincing imitation of a brain. The appearance that this deterministic machine gives of possessing free will is "mere sham"; but free will aside, it is "not altogether unreasonable" to describe a machine that "imitate[s] a brain" as itself being a brain. (As is well known, Turing advocates imitation as the basis of a test that "[y]ou might call . . . a test to see whether the machine thinks" [Turing et al. 1952: 466].)

In the course of the past four decades there have been a number of detailed suggestions for notional machines that, although completely deterministic, generate functions that cannot be generated by a universal Turing machine (Copeland and Sylvan 1999 is a survey; see also the references in my 1998d). These suggestions are of considerable interest to wide mechanists. They abundantly falsify the more expansive version of thesis M which abstracts from the issue of existence in the actual world. It remains an open empirical question whether or not the this-worldly version of thesis M is likewise false, and in particular whether the thesis is falsified by any deterministic mechanism. It is uncertain what Turing himself might have thought about this latter issue; if he ever discussed it, nothing appears to have survived.[24]

The proposition so important to Turing, that "[i]f any machine can appropriately be described as a brain, then any digital computer can be so described," is consistent with the view that the brain is computationally equivalent to, say, an O-machine whose oracle produces the values of the Halting function. For the foregoing argument of Turing's can readily be modified to cover this case: an appropriately programmed Turing machine will appear to an observer "who does not know the details of [its] construction" to behave in a perfectly brainlike fashion, and a machine that successfully imitates a brain can reasonably be said to be a brain. As Turing remarks elsewhere, the Turing machine will produce "an occasional wrong result," but this will hardly mark out the Turing machine from the brain (1947: 124).

Finally, what of the point that the behavior of any discrete-state machine with only a finite number of possible configurations can be simulated by a universal Turing machine? Does this undermine wide mechanism or provide a reason for saying that the historical mechanism of Descartes, Hobbes, La Mettrie et al. carried an implicit commitment to narrow mechanism? Not at all. For one thing, the mind may be some

form of continuous machine not simulable by Turing machine. But let us suppose for argument's sake that the mind is a discrete-state machine, and that, being situated in a world of bounded resources (time, energy, memory, and so on), the number of possible configurations that this machine can adopt is finite. In this case, each mind is simulable by a Turing machine equipped with a suitable look-up table, even if the table can be constructed only post hoc; but this provides no support for narrow mechanism. The crucial issue here is whether our cognitive architecture, abstracted from resource constraints, is best understood as being a generator of (one or more) Turing-machine-uncomputable functions, and the fact that the mind is simulable by Turing machine when certain resource constraints are operative says nothing either way. The wide mechanist stands firm on the claims that the empirical issue of how best to model the central mechanisms underlying cognitive performance is still pretty much completely open, and that there are no compelling reasons to believe that the model ultimately adopted will be selected from the narrow mechanist's artificially constrained space of models.

Acknowledgments

I am grateful to Stephen Gaukroger, Diane Proudfoot, and John Sutton for valuable discussion.

This chapter is an updated version of a paper that first appeared in the *Journal of Philosophy* (vol. 97, Jan. 2000).

Notes

1. Page references in this paragraph are to the appropriate volume of Cottingham, Stoothoff, and Murdoch (1984–1985).

2. Other distinctive but secondary claims were endorsed by various mechanists; often such claims were emphasized more by one thinker than by others, and in any case apply more to physical explanation than to physiological or psychophysiological explanation. These include the following (see McGuire 1972: 523): (1) Occult qualities are to be banished from explanations, which must be based on sensory experience in terms of clear and distinct ideas. (2) All natural phenomena arise from matter in motion, or matter and motion. (3) Compound bodies are composed of vortices (Descartes), centers of force (Leibniz), or microscopic corpuscles. (4) Metaphysical principles are to be integrated with experiment. (5) Nature is governed by immutable geometrical laws. (6) Regularities are to be expressed and explained in a mathematical manner. (7) Nature is to be conceived

dynamically in terms of motion, rather than statically in terms solely of size and shape. (8) An important feature of Cartesian mechanism was that matter be *inert,* in the sense that every change in the matter's motion is accounted for in terms of contact action. (In emphasizing the inertness, in this sense, of matter Descartes stood against "renaissance naturalism," i.e., the view that the material world is an essentially active realm [Gaukroger 1995: 146–152].)

3. Bechtel and Richardson speak aptly of the mechanist's twin heuristic strategies of *decomposition* and *localization* (1993: 23). The former heuristic seeks to decompose the activity of the system whose functioning is to be explained into a number of subordinate activities; the latter attributes these subordinate activities to specific components of the system.

4. Turing (1936, 1948: 5–6); Church (1937).

5. Lucas says in a recent retrospect of his argument: " 'Minds, Machines and Gödel' . . . was intended to show that minds were not Turing machines. . . . Having once got the hang of the Gödelian argument, the mind can adapt it appropriately to meet each and every variant claim that the mind is essentially some form of Turing machine" (1996: 103, 105).

6. See also Copeland (1998b,c; 1997; 1996; 1994; 1993, sections 5.5, 10.8).

7. Gandy (1980: 123–126) is one of the few writers to draw such a distinction.

8. Gandy (1980) uses the label "thesis M," but not the term "maximality thesis" (and his thesis M differs in certain respects from the maximality thesis).

9. Dennett (1995) appears simply to conflate the proposition that evolution is a "mindless, mechanical process" with the proposition that "evolution is an algorithmic process" (60, 75–76; see also 48–60 *passim*). (As is customary, he explicates the notion of an algorithm in terms of Turing machine activity: "Consider the set of all Turing machines—in other words, the set of all possible algorithms" [437].) This conflation appears to underlie his view that "algorithmic processes . . . have created the entire biosphere, ourselves included" (427).

10. A fuller account of O-machines may be found in my (1998a). For ease of exposition, the present account departs from Turing's own in various matters of detail.

11. In Turing's original exposition, these new fundamental processes produce the values only of Π^0_2 functions. In the subsequent technical literature, the notion of an O-machine has been widened to include fundamental processes that produce values of *any* function on the integers that is not Turing-machine-computable. I employ this extended notion here.

12. Chaitin has defined a number Ω that is analogous to, but not the same as τ. See, for example, his 1988.

13. *Pace* the intuitionists. Turing assumes a classical framework.

14. I take the claim that some entity *e* can be simulated by a Turing machine to mean that some Turing machine can pair any given descriptions of the stimuli impinging on *e* with either exact descriptions of *e*'s consequent behavior or descriptions that are accurate to any prespecified number of significant figures.

15. As is Dreyfus's similar claim that "*any* process which can be formalised so that it can be represented as a series of instructions for the manipulation of discrete elements can, at least in principle, be reproduced by [a universal Turing machine]" (1992: 72).

16. Turing concludes on the basis of neurophysiological evidence that the mind, if mechanical, is "not . . . a discrete-state machine" but a continuous machine (1950a: 451, 455). He sees no theoretical significance in this, however: "brains very nearly fall into this class [discrete-state machines], and there seems every reason to believe that they could have been made to fall genuinely into it, without any change in their essential properties" (1948: 6).

17. Hodges, for example, asserts this, attributing the view to Turing (1992: xvii; 1997: 34–36, 39).

18. Hodges (1997: 34) quotes extensively from the relevant pages of Turing (1950a) but fails to include the crucial words "discrete state machines . . . can be described by such tables *provided they have only a finite number of possible states*" (1950a: 440; my italics).

19. The arguments of the function generated by such an oracle are first call, second call, third call, . . . (or simply 1, 2, 3 . . .) and the first value of the function is the number that the oracle produces in response to the first call, and so on.

20. The date of writing of this paper is not known with certainty. It was presented on a radio discussion program called *The '51 Society*. Named after the year in which the program first went to air, *The '51 Society* was produced by the BBC Home Service at their Manchester Studio and ran for several years. (I am indebted to Peter Hilton for information.)

21. Newman in interview with Christopher Evans ("The Pioneers of Computing: an Oral History of Computing," Science Museum: London). Newman played an important part in Turing's intellectual life over many years. It was Newman who, in a lecture in Cambridge in 1935, introduced Turing to the concept that led directly to the Turing machine: Newman defined a constructive process as one that a *machine* can carry out. (ibid.) During the war, Newman and Turing both worked at the Government Code and Cypher School, Bletchley Park, where the two cooperated closely. It was Newman who initiated the electronic decryption project that culminated in the construction of Colossus, the first large-scale electronic digital computing machine (designed by the engineer T. H. Flowers). At the end of the war, Newman established the Royal Society Computing Machine Laboratory at the University of Manchester, where he introduced the engineers F. C. Williams and T. Kilburn to Turing's idea of a universal computing machine, and under Newman's guidance Williams and Kilburn built the first stored-program electronic digital computer (Copeland 1998a). In 1948, Newman appointed Turing Deputy Director of the Computing Machine Laboratory (there being no Director), and Turing remained at Manchester until his death in 1954.

22. In an early essay entitled "Nature of Spirit," possibly dating from Turing's undergraduate days, he wrote: "the theory which held that as eclipses etc. are predestined so were all our actions breaks down . . . We have a will which is able

to determine the action of the atoms probably in a small portion of the brain, or possibly all over it."

23. Turing devised a program that caused the Manchester computer to behave in an apparently partially random manner. When given a number the program would respond with another. Turing said "I would defy anyone to learn from these replies sufficient about the programme to be able to predict any replies to untried values" (1950a: 453).

24. Turing does endorse the thesis that results when the words "any machine" in the statement of thesis M are replaced with "any *calculating* machine," saying "[a] digital computer is a *universal* machine in the sense that it can be made to replace . . . any rival design of calculating machine" (1951b: 462). If he were pressed to make it clear exactly what is meant by "calculating machine," he would perhaps offer paradigm examples, as in his earlier (1948: 5–6): the Brunsviga and the NCR (popular desk calculating machines), the ENIAC (the electronic numerical integrator and computer), and so on. Or perhaps he would say, with greater generality, that a calculating machine is any machine that apes a human mathematician working with pencil and paper in accordance with a "rule of thumb" procedure (1948: 7). As previously remarked, it was in that manner that he explained the idea of an electronic computing machine in the opening paragraph of his *Programmers' Handbook*.

4

The Irrelevance of Turing Machines to Artificial Intelligence

Aaron Sloman

Editor's Note

Many computationalists (and their opponents) take it for granted that the notion of computation involved in computationalism is that of Turing-machine computability (or any of its extensionally equivalent formulations), without paying attention to the fact that there are significant differences between Turing machines and computers as used in cognitive science and AI research. Yet, according to Sloman, these differences make computers useful and Turing machines irrelevant to AI (and eventually computationalism). Sloman shows that computers, as built and used in AI research, are the result of a convergence of two historical developments: that of machines for automating various physical processes and that of machines for performing numerical calculations (i.e., performing abstract operations on abstract entities). It was the idea (involved in the latter kind of machine) of controlling abstract entities and processes that made a crucial step toward the development of computers as we know them today. Another important development was the use of machines with modifiable control functions that could be easily "programmed" to do different tasks. Whereas a machine's capacity to perform abstract operations can be studied from theoretical and practical viewpoints alike, Turing machines play a role only in theoretical investigations, although for AI research solely the practical aspects of computers and what operations they can perform matter. Historically, the possible ways in which different kinds of physical mechanisms could be used to control various physical processes were of the essence in modeling and understanding animal and human cognition. Such analyses, in turn, showed which features are needed for computers to be useful in modeling cognition, which then opened up new ways of thinking about the mind. Sloman compiles eleven such features, which arose from the development of machines manipulating physical entities as well as abstractions, and are highly relevant to the task of understanding, modeling, or replicating human or animal intelligence (and for the most part also animal brains). Yet, none of these features resulted from the concept "Turing machine." Examples of these features include the encoding of part of the system's behavior in substates of the system, the coupling of the system to the physical environment via transducers, or the handling of interrupts (i.e., the ability of the system to suspend and resume processes). Sloman points out that the unbounded resources of a Turing machine (as

embodied by its tape) are not only unrealistic assumptions about the nature of human minds, but are also not needed to understand the potentially unbounded competence of minds, since minds (and also computers) can implement (even if only partially) unbounded virtual machines without having to require an unbounded physical resource. In short, had Turing machines not been invented, contends Sloman, neither AI nor computationalism would have missed them.

1 Introduction

Many people think that our everyday notion of "computation," as used to refer to what computers do, is inherently linked to or derived from the idea of a Turing machine, or a collection of mathematically equivalent concepts (e.g., the concept of a recursive function, or the concept of a logical system). It is also often assumed, especially by people who attack AI, that the concepts of a Turing machine and Turing-machine computability (or mathematically equivalent notions) are crucial to the role of computers in AI and cognitive science.[1] For example, it is often thought that mathematical theorems regarding the limitations of Turing machines demonstrate that some of the goals of AI are unachievable.

I shall challenge these assumptions, arguing that although there is a theoretical, mathematically precise, notion of computation to which Turing machines, recursive functions, and logic are relevant, (1) this mathematical notion of computation and the associated notion of a Turing machine have little or nothing to do with computers as they are normally used and thought of, and (2) that although computers (both in their present form and in possible future forms if they develop) are extremely relevant to AI, as is computation defined as "what we make computers do," Turing machines are not relevant, and the development of AI did not depend even historically on the notion of a Turing machine.

In putting forward an alternative view of the role of computers and the idea of computation in AI, I shall try to clarify what it is about computers that makes them eminently suitable in principle, unlike previous man-made machines, as a basis for cognitive modeling and for building thinking machines, and also as a catalyst for new theoretical ideas about what minds are and how they work. Their relevance depends on a combination of features that resulted from two preexisting strands, or threads, in the history of technology, both of which started hundreds, or thousands, of years before the work of Turing and mathematical logicians.

The merging of the two strands and further developments in speed, memory size and flexibility were enormously facilitated by the production of electronic versions in the mid-twentieth century, not by the mathematical theory of computation developed at the same time or earlier.

A corollary of all this is that there are (at least) two very different concepts of computation: one of which is concerned entirely with properties of certain classes of formal structures that are the subject matter of theoretical computer science (a branch of mathematics), and another that is concerned with a class of information-processing machines that can interact causally with other physical systems and within which complex causal interactions can occur. Only the second is important for AI (and philosophy of mind).

Later I shall discuss an objection that computers as we know them all have memory limits, so that they cannot form part of an explanation of the claimed *infinite* generative potential of our thought and language, whereas a Turing machine with its unbounded tape might suffice for this purpose. Rebutting this objection requires us to explain how an infinite virtual machine can be implemented in a finite physical machine.

2 Two Strands of Development Leading to Computers

Two old strands of engineering development came together in the production of computers as we know them, namely (a) development of machines for controlling physical mechanisms and (b) development of machines for performing abstract operations, for example, on numbers.

The first strand included the production of machines for controlling both internal and external physical processes. Physical control mechanisms go back many centuries and include many kinds of devices, including clocks, musical-boxes, piano-roll mechanisms, steam engine governors, weaving machines, sorting machines, printing machines, toys of various kinds, and many kinds of machines used in automated or semi-automated assembly plants. The need to control the weaving of cloth, especially the need to produce a machine that could weave cloth with different patterns at different times, was one of the major driving forces for the development of such machines. Looms, like calculators and clocks, go back thousands of years and were apparently invented several times over in different cultures.[2]

Unlike the first strand, in which machines were designed to perform *physical* tasks, the second strand, starting with mechanical calculating aids, produced machines performing *abstract* operations on *abstract* entities, for example, operations on or involving numbers, including operations on sets of symbols to be counted, sorted, translated, etc. The operation of machines of the second type depended on the possibility of systematically mapping those abstract entities and abstract operations onto entities and processes in physical machines. But always there were two sorts of things going on: the physical processes such as cogs turning or levers moving, and the processes that we would now describe as occurring in a *virtual machine,* such as addition and multiplication of numbers. As the subtlety and complexity of the mapping from virtual machine to physical machine increased, it allowed the abstract operations to be less and less like physical operations.

Although the two strands were very different in their objectives, they had much in common. For instance, each strand involved both discrete and continuous machines. In the first strand, speed governors and other homeostatic devices used continuously changing values in a sensor to produce continuous changes in some physical output, whereas devices like looms and sorting machines were involved in making selections between discrete options (e.g., use this color thread or that one, go over or under a cross-thread). Likewise, some calculators used continuous devices such as slide rules and electronic analog computers, whereas others used discrete devices involving ratchets, the presence or absence of holes in cards, or electronic switches.[3]

Also relevant to both strands is the distinction between machines where a human operator is constantly involved (turning wheels, pushing rods or levers, sliding beads) and machines where all the processes are driven by motors that are part of the machine. Where a human is involved we can distinguish cases where the human is making decisions and feeding *control* information from cases where the human merely provides the *energy* once the machine is set up for a task, as in a music box or some mechanical calculators. If the human provides only energy it is much easier to replace the human with a motor that is part of the machine and needs only fuel.

In short, we can distinguish two kinds of autonomy in machines in both strands: energy autonomy and information or control autonomy.

Both sorts of autonomy developed in both physical control systems (e.g., in factory automation) and in machines manipulating abstract information (e.g., calculators).

At first, mechanical calculators performed fixed operations on small collections of numbers (e.g., to compute the value of some arithmetical expression containing a few numerical constants, each specified manually by a human operator). Later, Hollerith machines were designed to deal with large collections of numbers and other items such as names, job categories, names of towns, etc. This made it possible to use machines for computing statistics from census data. Such developments required mechanisms for automatically feeding in large sets of data, for instance on punched cards. Greater flexibility was achieved by allowing some of the cards to specify the operations to be performed on others, just as previously cards had been used to specify the operations to be performed by a loom in weaving.

This greater flexibility, in combination with the parallel development of techniques for feeding different sets of instructions to the same machine at different times (e.g., changing a weaving pattern in a loom), made it possible to think of machines that modified their own instructions while running. This facility extended control autonomy in machines, a point that was apparently understood by Babbage and Lovelace long before Turing machines or electronic computers had been thought of.

A natural development of numerical calculators was the production of machines for doing Boolean logic, inspired by ideas of George Boole in the nineteenth century (and Leibniz even earlier). This defined a new class of operations on abstract entities (truth values and truth tables) that could be mapped onto physical structures and processes. Later it was shown how numerical operations could be implemented using only components performing Boolean operations, leading to the production of fast, general-purpose, electronic calculating devices. The speed and flexibility of these machines made it possible to extend them to manipulate not only numbers and Boolean values but also other kinds of abstract information, for instance census data, verbal information, maps, pictorial information and, of course, sets of instructions, that is, programs.

These changes in the design and functionality of calculating machines originally happened independently of developments in metamathematics. They were driven by practical goals, such as the goal of

reducing the amount of human labor required in factories and in government census offices, or the goal of performing tasks with greater speed or greater reliability than humans could manage. Human engineering ingenuity did not have to wait for the development of mathematical concepts and results involving Turing machines, predicate logic or the theory of recursive functions, although these ideas did feed into the design of a subset of programming languages (including Lisp).

Those purely mathematical investigations were the main concerns of people like Frege, Peano, Russell, Whitehead, Church, Kleene, Post, Hilbert, Tarski, Gödel, Turing, and many others who contributed to the mathematical understanding of the *purely formal* concept of computation as some sort of philosophical foundation for mathematics.

Their work did not require the existence of physical computers. In fact some of the meta-mathematical investigations involved theoretical abstract machines that could not exist physically because they were infinite in size, or performed infinite sequences of operations.[4]

The fact that one of the important meta-mathematicians, Alan Turing, was also one of the early designers of working electronic computers simply reflected the breadth of his abilities: he was not only a mathematical logician but also a gifted engineer, in addition to being one of the early AI theorists (Turing 1950a; Hodges 1992).

3 Combining the Strands: Energy and Information

It was always inevitable that the two strands would merge, since often the behaviors required of control systems include numerical calculations, given that what to do next is often a function of internal or external measured values, so that action has to be preceded by a sensing process followed by a calculation. What had been learned about mechanical calculators and about mechanical control systems was therefore combined in new, extremely versatile information-based control systems, drawing the two strands together.

It is perhaps worth mentioning that there is a trade-off between the type of internal calculation required and the physical design of the system. If the physical design constrains behavior to conform to certain limits then there is no need for control signals to be derived in such a way as to ensure conformity, for example. Engineers have known for a long time

that good design of mechanical components of a complex system can simplify the task of the control mechanisms: it is not a discovery unique to so-called situated AI, but a well-known general principle of engineering, that one needs to consider the total system, including the environment, when designing a component. Another way of putting this is to say that some aspects of a control system can be compiled into the physical design.

Following that strategy leads to the development of special-purpose control mechanisms, tailored to particular tasks in particular environments. There are many exquisite examples of such special-purpose integrated designs to be found in living organisms. Evolution developed millions of varieties long before human engineers existed.

However, human engineers have also learned that there are benefits to the design of general-purpose, application-neutral computers, since these can be produced more efficiently and cheaply if numbers required are larger, and, more important, they can be used after their production in applications not anticipated by the designers. Evolution appears to have "discovered" a similar principle when it produced deliberative mechanisms, albeit in only a tiny subset of animal species. This biological development also preceded the existence of human engineers. In fact it was a precondition for their existence.

Understanding all this requires unpacking in more detail different stages in the development of machines in both historical strands. This reveals distinctions between different varieties of machines that help us to understand the significance of computers for AI and cognitive science.

Throughout the history of technology we can see (at least) two requirements for the operation of machines: energy and information. When a machine operates, it needs *energy* to enable it to create, change, or preserve motion, or to produce, change, or preserve other physical states of the objects on which it operates. It also needs *information* to determine which changes to produce, or which states to maintain. Major steps in the development of machines concerned different ways of providing either energy or information.

The idea of an energy requirement is very old and very well understood. The idea of an information requirement is more subtle and less well understood. I am here not referring to information in the mathematical sense (of Shannon and Weaver) but to an older, more intuitive notion of

information that could be called *control information,* since information is generally potentially useful in constraining what is done. I shall not attempt to define "information," because, like "energy," it is a complex and subtle notion, manifested in many forms, applicable to many kinds of tasks, and likely to be found in new forms in future, as previously happened with energy. So the concept is defined implicitly by the collection of facts, theories, and applications in which we use it: and therefore the concept is still developing.[5]

There are many subtleties involved in specifying what information is acquired, manipulated, or used by a machine (or an organism), especially as this cannot be derived unambiguously from the behavior of the organism or the nature of its sensors. For our purposes, however, we do not need to explain in more detail how to analyze precisely what control information is used by a machine. It suffices to acknowledge that some information is required, and that sometimes designers of a machine can explain what is happening. In the present context we note the fact that one difference between machines is concerned with where the energy comes from, and another concerns where the information comes from, discussed further below.

When a human uses a machine, the degree to which either the energy or the information comes from the human or from some other source can vary. Other types of variation depend on whether the energy or the information is provided ballistically or online, or in some combination of both.

Various kinds of machines such as water wheels, windmills, spring-driven or weight-driven machines, steam engines, electric motors, and many more are concerned with ways of providing energy that does not come from the user. Sometimes most of the control information comes from the user even if the energy does not. In many machines, such as cars, mechanical diggers, cranes, etc. the only energy required from the human user is that needed to convey the control information, for example, by turning wheels or knobs, pressing buttons or pedals, or pulling or pushing levers. Developments such as power-assisted steering or brakes, microswitches and other devices have reduced the energy required for supplying control information.

Sometimes the information determining what a machine should do is implicit in the physical structure of the machine and the constraints of

the situation in which it operates. For instance, a water wheel is built so that all it can do is rotate, though the speed of rotation is in part determined by the flow of water. In contrast, many machines are designed for use by humans who determine precisely what happens. The control information then comes from the user.

However, in general, some of the information will be inherent in the design of the machine and some will come from the environment. For instance, a windmill that automatically turns to face the wind gets its information about which way to turn from the environment.

Similar considerations apply to machines in the second strand: calculating machines. For example, the energy to move the beads of an abacus comes from the user, as does most of the control information determining which beads move when and where. However, some of the information comes from the changing state of the abacus, which functions in part as an extension of the user's memory. This would not be the case if at each step the abacus had to be disassembled and reconstructed with the new configuration of beads.

By contrast, in a primitive music box, a human may continuously provide *energy* by turning a handle while all the *control information* determining which sounds to produce next come from something in the music box, for example, a rotating cylinder or disc with protruding spokes that pluck or strike resonating bars of different lengths. The only control the human has is whether to continue or to stop, or perhaps whether to speed up the music or slow it down, depending on the construction of the music box. Some music boxes may also have a volume or tone control that can be changed while the music is playing.

Both the energy and the information required to drive a machine may be provided by a user in either an *online* or a *ballistic* fashion. If a music box accepts changeable cylinders with different tunes, the user will have control, but only ballistic control: by setting the total behavior at the beginning. Likewise energy may be provided in a ballistic fashion, if the music box is wound up and then turned on and left to play. At the opposite extreme, playing a violin or wind instrument requires exquisite online provision of both energy and information.

The combined online provision of both energy and control information is characteristic of tools or instruments that allow humans to perform actions that are difficult or impossible for them to do unaided, because

of limitations of strength, height, reach, or perceptual ability, or because body parts are the wrong size or shape (e.g., tweezers are often used where fingers are too big or the wrong shape), or because we cannot make the same sound as the instrument. In such cases the user is constantly in control during the operation of such a machine, providing both *energy* but also the *information* required to guide or manipulate the tool. In other machines most of the energy may come from some other source, while the human provides only the energy required to operate control devices, for instance when power-assisted steering reduces the amount of energy required from the user without reducing the amount of information provided. In other words, the user is still constantly specifying what to do next.

Ballistic information provision can vary in kind and degree. In the case of the music box or machine driven by punched cards the sequence of behaviors is totally determined in advance, and then the machine is allowed to run through the steps. However, in a modern computer, and in machines with feedback control mechanisms, some or all of the behavior is selected on the basis of some tests performed by the machine even if it is running a program that was fully specified in advance. If the tests and the responses to the tests are not predetermined but rather produced by some kind of learning program, or by rules that cause the initial program to be modified in the light of which events occur while it is running (like an incremental compiler used interactively), then the ballistic control information provided initially is less determinate about the behavior. It may rigidly constrain sets of possible options, but not which particular options will be selected when.

If the initial information provided to the machine makes possible a large collection of actions but is not specific about the order in which they should be performed, leaving the machine to make selections on the basis of information acquired while behaving, then the machine is to some extent autonomous. The degree and kind of autonomy will vary.[6]

For many types of applications, the control functions of the machine could be built directly into its architecture, because it repeatedly performed exactly the same sort of task, for example, telling time, playing a tune. This was rigid ballistic control.

For other applications, such as weaving cloth with different patterns, it was desirable not to have to assemble a new machine for each task.

This required a separation of a fixed reusable physical architecture for performing a class of tasks and a variable behavioral specification that could somehow harness the causal powers of the architecture to perform a particular task in that class.

For some of the earlier machines, the variable behavior required continuous human intervention (e.g., playing a piano, operating a loom, manipulating an abacus), that is, only online control could produce variable results. Later, in some cases, it was possible to have various physical devices that could be set manually at the start of some task to produce a required behavioral sequence, and then reset for another task, requiring a different sequence of behaviors. This was variable ballistic control. This might require setting levers or cog-wheels to some starting position, and then running the machine. In the case of the earliest electronic control systems, this meant setting switches, or altering electric connections before starting the process.

At the beginning of the nineteenth century, Jacquard realized that the use of punched cards could make it much easier to switch quickly between different behavioral sequences for looms. The cards could be stored and reused as required. A similar technique used punched rolls of paper, as in player pianos. These mechanisms provided easily and rapidly specified variable ballistic control.

Later, the same general idea was employed in Hollerith card-controlled machines for analyzing census data, and paper-tape-controlled data-processing machines. In these cases, unlike looms, some of the ballistic control was concerned with selection of *internal* action sequences.

The alteration of such physically encoded instructions required human intervention, for example, feeding in punched cards, or piano rolls, or in the case of some music boxes, replacing a rotating disc or cylinder with metal projections.

In Babbage's design for his "analytical engine," the use of conditionals and loops allowed the machine to decide for itself which collection of instructions to obey, permitting great flexibility. However, it was not until the development of electronic computers that it became feasible to produce computers that, while running, could create new programs for themselves and then obey them.[7]

Machines programmed by means of punched cards had reached considerable sophistication by the late nineteenth and early twentieth century,

long before electronic computers, and long before anyone had thought of Turing machines, recursive function theory, or their mathematical equivalents.

The electronic technology developed during the twentieth century allowed faster, more general, more flexible, more reliable machines to be produced, especially after the invention of transistors allowed the replacement of electromechanical relays and vacuum tubes. The advent of randomly addressable memory facilitated the development of machines that could not only rapidly select arbitrary instructions to follow, but could also change their own programs easily at run time.

The process of development of increasingly sophisticated information processing systems was accelerated during the 1970s onward, both by advances in materials science and electronic engineering, and also by the rapid evolution of new computer-assisted techniques for designing new machines and computer-controlled fabrication techniques.

In other words, the production of new, improved machines for controlling physical processes accelerated the production of even better machines for that purpose. Some of this depended crucially on the second strand of development: machines for operating on abstract entities, such as numbers. The ability to operate on abstract entities was particularly important for machines to be able to change their own instructions, as discussed below. Developments in electronic technology in the second half of the twentieth century facilitated construction of machines that could alter their internal control information while running. However, the importance of this had at least partly been understood earlier: it did not depend on the idea of Turing machines, which had this capability, but only in a particularly clumsy form.

3.1 Toward More Flexible, More Autonomous Calculators

Numerical calculators, like machines for controlling physical processes, were invented many centuries ago and evolved toward more and more sophisticated and flexible machines.

Only recently have they achieved a degree of autonomy. The earliest devices, like the abacus, required humans to perform all the intermediate operations to derive a result from some initial state, whereas later calculators used increasingly sophisticated machinery to control the operations

that transformed the initial state, representing a problem, to a state where the solution could be read off the machine (or in later systems printed on paper, or punched onto cards).

In the earliest machines, humans had to provide both the *energy* for making physical changes to physical components (e.g., rotating cogs) and the *information* about what to do next. At a later date it sufficed for a human to initialize the machine with a problem and then provide the energy (e.g., by turning a handle) in a manner that was neutral between problems and did not feed additional information into the machine. Eventually even the energy for operation did not need to be supplied by a human, as the machines began to use electrical power from mains or batteries.

3.2 Internal and External Manipulations

In all these machines we can, to a first approximation, divide the processes produced by the machine into two main categories: *internal* and *external*. Internal physical processes include manipulation of cogs, levers, pulleys, strings, etc. The external processes include movements or rearrangements of various kinds of physical objects, for example, strands of wool or cotton used in weaving, cards with information on them, lumps of coal to be sorted according to size, parts of a musical instrument used to produce sounds, objects being assembled on a production line, printing presses, cutters, grinders, the things cut or ground, etc.

If the internal manipulations are merely part of the process of selecting which external action to perform or part of the process of performing the action, then we can say that they are *directly subservient* to external actions.

However, internal actions that are part of a calculation are an especially important type of action, for they involve abstract processes, as discussed previously. Other abstract internal processes involve operations on nonnumeric symbols and structures, such as words, sentences, encrypted messages, arrays, lists, trees, networks, etc. A particularly important type of internal action involves changing or extending the initially provided information store. This gives machines considerable additional flexibility and autonomy. For instance, they may end up performing actions that were neither foreseen nor provided by the designer.

3.3 Practical and Theoretical Viewpoints

The requirement that a machine be able to perform abstract operations can be studied from two viewpoints. The first is the practical viewpoint concerned with producing machines that can perform useful specific tasks, subject to various constraints of time, memory requirements, cost, reliability, etc. From this viewpoint, it may be sensible to design different machines for different tasks, and to give machines the powers they need for the class of tasks to which they will be applied. For this reason there are specialized machines for doing integer operations, for doing floating point operations, for doing operations relevant to graphical or acoustic applications, for running neural nets, etc. It is to be expected that this variety of machines that can be combined within computers and other kinds of machinery will continue to grow.

The second, more theoretical viewpoint is concerned with questions like:

- What is the *simplest* machine that can perform a certain class of tasks?
- For a given type of machine, what is the class of tasks that it can perform?
- Given two machines M_1 and M_2, is one of them more general, for example, able to perform all the tasks of the other and more besides?
- Given two machines M_1 and M_2, are they equivalent in their capabilities, for example, can each provide the basis for an implementation of the other?
- Is there a machine for performing abstract asks (e.g., mathematical calculations, or logical inferences) that is *most general* in the sense that it is at least as general as any other machine that can perform abstract tasks?

From the theoretical viewpoint, Turing machines are clearly of great interest because they provide a framework for investigating some of these questions (though not the only framework). If AI were concerned with finding a single most general kind of information-processing capability, then Turing machines might be relevant to this because of their generality. However, no practical application of AI requires total generality, and no scientific modeling task of AI (or cognitive science) requires total generality, for there is no human or organism that has completely general capabilities. There are things that chimps, or even bees, can do that humans cannot, and vice versa.

The mathematical applications of the idea of a Turing machine did not depend on the actual existence of such machines: they were concerned with a purely formal concept of computation. However, it is possible in principle to build a Turing machine, although any actual physical instance must have a finite tape if the physical universe is finite.

We can now see Turing machines as just one of a class of machines that are capable of performing either the task of controlling a physical system or of performing abstract operations, or of using one to do the other. Their most important single characteristic is that they have an unbounded tape, but that is possible only if they are treated as mathematical constructs, for physical machines will always have a bounded tape.

However, that unique feature cannot be relevant to understanding human or animal brains, since they are finite in any case. No human being has a memory that has unlimited capacity like a Turing machine's tape. Even if we include the external environment, which can be used as an extension of an individual's memory, anyone who has written or bought many books or who has created many computer files knows that as the total amount of information one records grows the harder it becomes to manage it all, to find items that are relevant, and even to remember that you have some information that is relevant to a task, let alone remember where you have put it. There is no reason to believe that humans could manage unlimited amounts of information if provided with an external store of unlimited capacity, quite apart from the fact that we live only for a finite time.

In a later section, we'll consider the argument that these limitations of human beings are merely *performance* limitations, and that we really do have a type of infinite, or at least unbounded, *competence*. It will be shown that analogous comments can be made about conventional computers, which do not have the unbounded memory mechanism of a Turing machine.

Having previously shown that the development of computers owed nothing to the idea of a Turing machine or the mathematical theory of computation, we have now given a negative answer to the question of whether Turing machines, viewed as simply a special type of computer, are required for modeling human (or animal) minds because the unbounded tape of a Turing machine overcomes limitations of more conventional computers.

Turing machines, then, are irrelevant to the task of explaining, modeling, or replicating human or animal intelligence, though they may be relevant to the mathematical task of characterizing certain sorts of esoteric unbounded competence. However, computers have features that make them relevant to modeling intelligence and that do not depend on any connection with Turing machines, as will now be shown.

4 Computers in Engineering, Science, and Mathematics

The features of computers that grew out of the two strands of development made them powerful and versatile tools for a wide variety of tasks, which can be loosely classified as engineering, science, and mathematics. The notion of a Turing machine and related logical and mathematical notions of computation are only indirectly relevant to most of these. In fact, as explained above, many of the applications were being developed before the time of Turing. AI overlaps with all of these application areas in different ways. I shall make a few comments on the relevance of computers to all these areas before going on to a more detailed analysis of the relevance of computers to AI and cognitive science. However, it will help to start with an analysis of their general relevance to engineering and science.

4.1 Engineering and Scientific Applications

Most of the features of calculating and controlling engines (e.g., the ability to manipulate physical objects and abstract entities such as numbers or symbols) are equally relevant to a variety of application domains: industrial control, automated manufacturing systems, data-analysis and prediction, working out properties of complex physical systems before building them, information management in commercial, government, and military domains, many varieties of text processing, machine interfaces to diverse systems, decision support systems, and new forms of communication. These applications use different aspects of the information manipulating capabilities described above, though with varying proportions and types of ballistic and online control, and varying proportions of physical manipulation and manipulation of abstract entities. None of this had anything to do with Turing machines.

In addition to practical applications, computers have been enormously relevant to science construed as the attempt to understand various aspects of reality. For instance, they are used:

- to process numerical and other data collected by scientists;
- to control apparatuses used in conducting experiments or acquiring data;
- to build working models capable of being used as explanations; and
- to make predictions that can be used to test scientific theories.

For some of these uses, the ability of computers to control devices that manipulate physical objects is particularly relevant; for others, the ability to manipulate abstractions such as numbers, laws, hypotheses is more relevant.

4.2 Relevance to AI

The very features that made computers relevant to all these engineering applications, and to science in general, also make them relevant to both the scientific aims of AI and the engineering aims.

The scientific aims of AI include understanding general features of both natural and artificial behaving systems, as well as modeling and explaining a wide variety of specific naturally occurring systems, for instance, different kinds of animal vision, different kinds of animal locomotion, different kinds of animal learning, etc.

Since the key features of such natural systems include both being able to manipulate entities in the environment and being able to manipulate abstract entities, such as thoughts, desires, plans, intentions, theories, and explanations, the combined capabilities of computers made them the first machines suitable for building realistic models of animals.

Moreover, the tasks of designing, extending, and using these capabilities of computers led to the development of a host of new formalisms and concepts relevant to describing, designing, and implementing information-processing mechanisms. Many of these are relevant to the goals of AI and will be described below.

The engineering aims of AI include using computers to provide new sorts of machines that can be used for practical purposes, whether or not they are accurate models of any form of natural intelligence. These engineering aims of AI are not sharply distinguished from other types of

applications of computers that are not described as AI. Almost any type of application can be enhanced by giving computers more information and more abilities to process such information, including the ability to learn from experience. In other words, almost any computer application can be extended using AI techniques.

It should now be clear why computers are relevant to all the different subdomains of AI that deal with specific aspects of natural and artificial intelligence, such as vision, natural language processing, learning, planning, diagrammatic reasoning, robot control, expert systems, intelligent internet agents, distributed intelligence, etc.—they all involve some combination of control of physical processes and abstract information manipulation processes, tasks for which computers are better than any preexisting type of machine.

It is noteworthy that computers are used by supporters of all the rival "factions" of AI, each of which adopts different sorts of designs, such as rule-based systems, logicist systems, neural nets, evolutionary computation, behavior-based AI, dynamical systems, etc.

Thus there is no particular branch of AI or approach to AI that has special links with computation: they all do, although they may make different use of concepts developed in connection with computers and programming languages. In almost all cases, the notion of a Turing machine is completely irrelevant, except as a special case of the general class of computers. Moreover, Turing machines are not so intrinsically relevant as machines that are designed from the start to have interfaces to external sensors and motors with which they can interact online, unlike Turing machines, which at least in their main form are totally self-contained and are designed primarily to run in ballistic mode once set up with an initial machine table and tape configuration.

4.3 Relevance to Mathematics

The relevance of computers to mathematics is somewhat more subtle than their relevance to other scientific and engineering disciplines. There are at least three types of development that link mathematics and computing:

(a) *More mathematics:* using abstract specifications of various kinds of (abstract) machines and the processes they can support, in order to define one or more new branches of mathematics, for example, the study of complexity, computability, compressibility, various properties of algo-

rithms, etc. This is what much of theoretical computer science is about. (Some of this investigates machines that could not be built physically, e.g., infinite machines, and types of machines that might be built but have not, e.g., inherently probabilistic machines.)

(b) *Metamathematics:* using an abstract specification of a type of machine as an alternative to other abstract specifications of types of mathematical objects and processes (recursive functions, Post productions, axiomatic systems, etc.), and then exploring their relationships (e.g., equivalence), possibly to clarify questions in the philosophy of mathematics.

(c) *Automatic theorem proving or checking:* using computers as tools to help in the discovery or proof of theorems or in searches for counterexamples. This process can be more or less automated. At one extreme, computers are programmed to do large numbers of well-defined but tedious operations, for example, examining very large sets. At another extreme, the computer may be fairly autonomous, taking many decisions about which steps to try in a context-sensitive manner and possibly as a result of learning from previous tasks. AI work on theorem proving tends toward the latter extreme. It may also allow human interaction, such as the communication that happens between human mathematicians when they collaborate or when one teaches another. This sort of mathematical application could build on general AI research on intelligent communication.

Although mathematical explorations of types (a) and (b) involve ideas about computation, it often does not matter whether physical computers exist or not, for they are not needed in those explorations. Many of the important results, for instance Gödel's undecidability result, were achieved before working computers were available. (Quantum computers might also be relevant to mathematical investigations of types (a) and (b) even if it turns out they are impossible to build as practically useful physical devices.) By contrast, work of type (c) depends on the use of working computers.

The distinction between (a) and (b) is not yet very clear or precise; in fact, (a) subsumes (b). Neither is there a very sharp division between the metamathematical use of the notion of computation in (b) and the AI uses in connection with designing theorem provers, reasoners, etc.

Ideas about Turing machines and related theoretical "limit" results on computability, decidability, definability, provability, etc. are relevant to all these kinds of mathematical research, but they are marginal or

irrelevant in relation to most aspects of the scientific AI goal of trying to understand how biological minds and brains work, and also to the engineering AI goals of trying to design new useful machines with similar (or greater) capabilities. The main relevance of the limit results arises when researchers set themselves goals that are known to be unachievable, such as trying to design a program that will detect infinite loops in any arbitrary program.

The metamathematical ideas developed in (b) are relevant to the small subset of AI that is concerned with *general* (logical) reasoning capabilities or modeling *mathematical* reasoning.

By contrast, the new mathematical techniques of type (a), which were developed for analyzing properties of computational processes such as space and time complexity and for analyzing relationships between specifications, designs, and implementations, are all equally relevant both to AI and to other applications of computers.

One important feature of Turing machines for mathematical or metamathematical research of types (a) and (b) is their universality, mentioned previously. By showing how other notions of mathematical reasoning, logical derivation, or computation as an abstract mathematical process could all be mapped onto Turing machines, it was possible to demonstrate that results about mathematical limitations of Turing machines could not be overcome by switching to any of a wide range of alternative formalization. It also meant that analyses of complexity and other properties of processes based on Turing machines could be carried over to other types of process by demonstrating how they were implementable as Turing machine processes.[8]

This kind of mathematical universality may have led some people to the false conclusion that any kind of computer is as good as any other provided that it is capable of modeling a universal Turing machine. This is true as a mathematical abstraction, but it is misleading or even false when considering problems of controlling machines embedded in a physical world.

The universality of Turing machines was mentioned by Turing (1950a) as a reason for not discussing alternative digital mechanisms. In part that was because he was considering a question-answering task for which there were no time constraints, and where adding time constraints would

produce no interesting differences, since only qualitative features of the behavior were of interest.

Human intelligence, however, is often *precisely* concerned with finding good solutions to problems quickly, and speed is central to the success of control systems that manage physical systems embedded in physical environments. Aptness for their biological purpose, and not theoretical universality, is the important characteristic of animal brains, including human brains. What those purposes are and what sorts of machine architectures can serve those purposes are still open research problems (which I have discussed elsewhere), but it is clear that time constraints are very relevant to biological designs: speed is more biologically important than theoretical universality.

I have introduced these mathematical applications of ideas of computation in order to get them out of the way, and in order to provide a possible explanation for the widespread but mistaken assumption that notions such as Turing machines or Turing computability are central to AI. (This is not to deny that Turing was important to AI as an outstanding engineer who made major contributions to the development of practical computers. He was also important as one of the earliest AI theorists.)

4.2 Information-Processing Requirements for AI

For the mathematical and metamathematical investigations mentioned above, the formal notions of computations were central. By contrast, for the nonmathematical, scientific, and engineering goals of AI, the important point, which was already clear by about the 1950s, was that computers provided a new type of *physically implementable* machine with a collection of important features discussed in previous sections and analyzed in more detail below.

These features were not defined in relation to recursive functions, logic, rule formalisms, Turing machines, and so on but rather had to do with using machines to produce and control sophisticated internal and external behavior with a speed and flexibility that was previously impossible for man-made machines. Various combinations of these abilities were to be found in precursors to modern computers, but many of these mathematically important features of Turing machines were irrelevant to animal brains.

However, I shall identify two related features that are relevant, namely (i) the ability to chunk behaviors of varying complexity into reusable packets, and (ii) the ability to create and manipulate information structures that vary in size and topology.

A Turing machine provides both of these features to an unlimited degree, but it depends on a linear, indefinitely extendable tape, whose speed of use is inherently decreased as the amount of information thereon increases. However, for an animal or robot mind, it is not clear that *unlimited* size of chunks or variability of structure would be useful, and the cost of providing it may be excessive. By contrast, computers with random access memory provide uniform speed of access to a *limited* memory. Brains probably do something similar, though they differ in the details of how they manage the trade-off.

Although humans do not have the same generality as Turing machines in their mathematical and symbolic reasoning powers, nevertheless we do have certain kinds of generality and flexibility, and I shall try to explain below how computers, and also brains, can provide them. Turing machines provide much more generality but do so in a fashion that involves such a heavy speed penalty in any working physical implementation, because of the need for repeated sequential traversal of linear tapes, that they seem to be worthless for practical purposes. It appears that brains, like computers, sacrifice total generality in favor of speed in a large class of behaviors.

4.5 Does AI Require the Use of Working Machines?

A possible source of confusion is the fact that for some of the theoretical/scientific purposes of AI (cognitive science) the actual construction of computers did not matter except as a catalyst and test-bed for ideas: the most important effect of development of computers for advances in AI and cognitive science was in the generation of ideas about how information-manipulating engines might be implemented in physical mechanisms. This makes scientific AI superficially like metamathematics, but the similarity is deceptive, since the goals are different.

For the engineering purposes of AI, working physical machines are, of course, required. They are also needed as an aid to AI theorists, since the models being considered have grown increasingly complex, and too

difficult to run in our heads or on paper. But that is like the relevance of computers to theoretical chemistry, astronomy, etc. Computers are tools for managing complex theories. This has nothing specific to do with cognitive science or AI.

5 Eleven Important Features of Computers (and Brains)

More important than the use of computers as cognitive aids to analyzing complex theories was the way we came to understand the features needed in computers if they were to be useful. Those features gave us new ideas for thinking about minds.

However, those features are not specific to what we now call computers: animal brains had most of these features, or similar features, millions of years earlier, and probably lots more that we have not yet thought of, and will later learn to emulate, using either computers or new kinds of machines.

There are (depending on how we separate them out) about eleven major features, which I shall list below. Features F1 to F6, which I have labeled "primary features," are typically built in to the digital circuitry and/or microcode of computers and have been common to low-level virtual machine architectures since the 1960s (or earlier), though details have changed.

The remaining features, labeled "secondary" below, depend very much on software changing the higher level virtual machine implemented in the basic lower level machine. The need for those extra features to be added was driven both by AI requirements and by other software engineering requirements.[9]

Besides the primary and secondary features of computers as control systems listed below, there are further features that are added through the design and implementation of various high-level languages, along with compilers, interpreters, and software development tools. These extend the variety of virtual machines available. However, these split into several different families suited to different application domains and will not be discussed here.

F1. State variability: having very large numbers of possible internal states, and even larger numbers of possible state transitions

Both of these—having large numbers of possible internal states and state transitions—are consequences of the fact that computers have large numbers of independently switchable components, like brains. *N* two-valued components can be in 2^N possible states, and can support the square of that number of possible one-step state transitions: this gives huge flexibility in coping with varied environments, goals, and forms of learning.

Of course, it is not enough that there are large numbers of components that can change their states. That is equally true of thunderclouds. It is also important that the switchable components are controlled in a principled way, depending on the system's current tasks and the available information. Further, it is also important that those substates can be causally connected in a principled way to various effects, both within the machine and in external behavior. This depends on the next three features.

F2. Laws of behavior encoded in substates: having laws of behavior determined by parts of the internal state

The behavior of any physical object is to some extent controlled by its substates: for example, how a rock or a bean-bag rotates if thrown into the air will depend on its shape and the distribution of matter within it, which may change in the bean-bag. However, computers have far more independently switchable persistent substates, and their effects can be made more global: a CPU is an influence-amplifier. All this depends on the fact that a stored program typically includes components that have procedural semantics and whose execution generates state transitions in accordance with the program, for example, using sequencing, conditionals, jumps, procedure invocation (if supported; see secondary feature F7 below), etc. The effects can propagate to any part of the system.

It is important, however, that not all stored information states are concurrently active, as occurs when a large number of different forces are simultaneously acting on a physical object whose behavior is then determined by the resultant of those forces. In computers, as in brains, different pieces of stored information can become active at different times. This is related to the point about conditional instructions (see below). It is also connected with Ryle's emphasis on the *dispositional* properties of minds (Ryle 1949). Minds, like computers, have many dormant dispositions that are activated when conditions are appropriate. This fine-grained

control of a very large number of distinct capabilities is essential for most biological organisms.

F3. Conditional transitions based on Boolean tests

For many physical systems, behavior changes continuously through additive influences of multiple continuously varying forces. The behavior of such systems can typically be represented by systems of differential equations. In contrast, there is a small subset of physical systems, including some organisms and some machines, in which behaviors switch between discrete alternatives on the basis of Boolean (or more generally discrete-valued) tests. The laws of such a system may include conditional elements, with forms like

if X then do A else do B,

possibly implemented in chemical, mechanical, or electronic devices. Systems controlled by such conditional elements can easily be used to approximate continuous dynamical systems, as is done every day in many computer programs simulating physical systems. However, it is hard to make the latter simulate the former—it requires huge numbers of carefully controlled basins of attraction in the phase space. One way to achieve this is to build a machine with lots of local dynamical systems that can be controlled separately—that is, a computer!

F4. Referential "read" and "write" semantics

If the behavior of a system is to be controlled in a fine-grained way by its internal state, the active elements of the system need some way of accessing or interrogating relevant parts of the internal state, for instance in testing conditions or selecting control signals (instructions) to become active. Likewise, if a system is to be able to modify its internal state in a controlled way so as to influence future behavior, it needs to be able to modify parts of itself so that they can be interrogated later.

In principle, there are many ways this ability to interrogate or change specific components can be implemented. In computers it involves allowing bit patterns in memory and in address registers to be interpreted, by the machine, as referring to other parts of the machine: a primitive kind of "referential semantics"—discussed further in Sloman (1985, 1987), where it is argued that this can form the basis for many other kinds of semantic capabilities. In early computers the reference was to

specific *physical* parts of the system. Later it was found more useful to support reference to *virtual* machine locations, whose physical implementation could vary over time.

F5. Self-modifying laws of behavior

In some machines and organisms, the features listed previously can be combined in such a way that the machine's laws of behavior (i.e., the stored programs) can be changed while the machine is running by changing the internal state (e.g., extending or modifying a stored program). Such changes involve internal behaviors based on features F2, F3, and F4. Such self-modification can be useful in many ways, including long-term changes of the sort we call learning and short-term changes where what is sensed or perceived at a point in time can be stored long enough to be used to modify subsequent actions. I suspect that the full variety of self-modifications in animals, also required for sophisticated robots, has not yet been appreciated, for instance changes that alter the virtual machine architecture of a human during development from infancy to adulthood.

F6. Coupling to environment via physical transducers

I have implicitly been assuming that some parts of a system can be connected to physical transducers so that both sensors and motors can be causally linked to internal state changes.

If external sensors asynchronously change the contents of memory locations, that allows the above "read" capabilities to be the basis for perceptual processes that control or modify actions. Likewise, if some locations are linked through transducers to motors, then "write" instructions changing those locations can cause signals to be transmitted to motors, which is how internal information manipulation often leads to external behavior. Thus sensors can write information into certain memory locations that can then change subsequent internal behavior, and some of the memory locations written to by the internal processes can cause signals to be sent to external motors. This implies that the total system has multiple physical parts operating asynchronously and concurrently.[10] Perception and action need not be restricted to "peephole" interactions through very narrow channels (e.g., transmitting or receiving a few bits at a time). Where large numbers of input transducers operate in

parallel, it is possible for different perceptual patterns at different levels of abstraction to be detected and interpreted: multiwindow perception. Likewise, concurrent multiwindow actions can occur at different levels of abstraction.

Interesting new possibilities arise if some of the transducers asynchronously record not states of the external environment but *internal* states and processes, including those in physical and virtual machines. In computers this plays an important role in various kinds of error checking, memory management, detection of access-violations, tracing, and debugging. See also the section on self-monitoring, below.

I believe all of the above features of computers were understood at least intuitively and implemented at least in simple forms by the late 1950s and certainly in the 1960s. None of this depended on knowledge of Turing machines.

We now turn to "secondary features" of computers. These are usually not inherent in the hardware designs, but can be added as virtual machines implemented on the basis of the core features F1 to F6. (In some computers they are given hardware support.)

Some of these features were needed for general programming convenience (for example, enabling chunks of code to be shared between different parts of the same program), and some were needed because computers were interacting with an unpredictable environment (e.g., a human or some other machine). None of these features was required *only* for AI purposes. In other words, they were all part of the continuing general development of the two main historical strands: producing more sophisticated, flexible, and autonomous systems for controlling physical machinery, and producing more sophisticated mechanisms for manipulating abstract structures.

F7. Procedure control stack: the ability to interrupt a process, suspend its state, run some other behavior, then later resume the original suspended process.

This feature facilitated programs with reusable modules that could be invoked by other modules to which they would automatically return when complete. Later versions allowed the reusable modules to be parameterized, that is, to have different "local" data on different invocations (unlike GOSUB and RETURN in BASIC). This allowed recursive

procedures to perform slightly different tasks on each recursive activation. This feature, along with feature F10 below (support for variable-length information structures), was particularly important in supporting some of the descriptive, nonnumeric, AI techniques discussed in Minsky (1961).

Eventually all this was facilitated in electronic computers by hardware or microcode support for a "control stack," that is, special memory operations for suspended process descriptors, plus mechanisms for pushing and popping such process descriptors. This depends on the saved process states being specifiable by relatively small state descriptors that can be rapidly saved and restored. It would be hard to do that for a neural net, or a mechanical loom. It can be done by mechanical devices, but it is far easier to implement in electronic mechanisms.

It might be done by a *collection* of neural nets, each implementing one process, where only one can be in control at a time. This presupposes a fixed process topology, unless there is a supply of spare nets that can be given new functions. The "contention scheduling" architecture (Cooper and Shallice 2000) is something like this.

F8. Interrupt handling

The ability to suspend and resume processes also allowed a system to respond to external interrupts (new sensory input) without losing track of what it had been doing previously. This kind of asynchronous interrupt is unlike the previous case where control is transferred by an explicit (synchronous) instruction in the program that is to be suspended. Asynchronous interrupts can be supported by software polling in an interpreter. Faster, lower-level support built in to the computer's hardware mechanisms reduces the risk of losing data and sometimes simplifies software design, but the key idea is the same.

F9. Multiprocessing

On the basis of extensions of the previous mechanisms it became fairly easy to implement multiprocessing systems, which could run many processes in parallel on a single CPU in a time-shared fashion, with interrupts produced at regular intervals by an internal clock instead of (or in addition to) interrupts generated by external events. This requires larger contexts to be saved and restored, as processes each with their own control stacks are switched in and out.

Such multiprocessing allows the development of virtual machine architectures that permit variable numbers of concurrent, persistent, asynchronously interacting processes, some of them sharing memory or subroutines with others. It provides more flexibility than could be achieved by wiring together a collection of computers that were each running one process, since then the maximum number of processes would be fixed by the number of computers.

Multiprocessing virtual machines allow new kinds of experiments with mental architectures containing variable numbers of interacting virtual machines, including various sorts of perceptual, motivational, learning, reactive, deliberative, planning, plan execution, motor-control, and self-monitoring processes. This is a far cry from the notion of AI as the search for "an algorithm" (as discussed by Searle 1980 and Penrose 1989, and criticized in Sloman 1992).

In the early days of AI, research focused on algorithms and representations, whereas in the last decade or so, there has been a growing emphasis on complete systems with architectures that include many submechanisms running concurrently. One consequence is that different algorithms can interact in unexpected ways. This is important for the design of a fully functioning intelligent robot (or animal) with multiple sensors, multiple motors, in a changing and partly unpredictable environment, with a variety of more or less independent motive generators operating asynchronously under the influence of both internal processes and external events (Beaudoin 1994).

However, I don't think that many serious experiments of this sort were done by AI researchers before the 1980s. Such experiments were drastically impeded by speed and memory limitations at that time. (Examples were some of the blackboard architectures. My POPEYE system in the late 1970s [Sloman 1978] was an attempt at a visual architecture with different concurrent mutually interacting layers of processing, but it was against the spirit of the time, since the influence of Marr [1982] was dominant.)

F10. Larger virtual data chunks

Another feature of computers that became important both for AI and for other purposes (e.g., databases, graphic design) was the ability to treat arbitrarily large chunks of memory as "units." There were various ways

this could be done, including reserving a collection of contiguous (physical or virtual machine) locations as a single "record" or "array" with some explicit information specifying the beginning and the length so that all the memory management and memory access mechanisms respected that allocation.

More subtle and flexible mechanisms used list-processing techniques implemented in software (though there have been attempts to provide hardware support to speed this up). This allowed the creation of larger structures composed of "chained" pairs of memory locations, each containing some data and the address of the next link in the chain, until a "null" item indicated that the end had been reached. This allowed new links to be spliced into the middle of a chain, or old ones deleted, and also permitted circular chains (of infinite length).

The full power of this sort of mechanism was first appreciated by McCarthy and others interested in logic and recursive functions. But it is a generally useful technique for many purposes that have nothing to do with AI or cognitive science, and was bound to be reinvented many times.[11]

Thus, whereas machine hardware usually treats a fixed size bit-pattern as a chunk (e.g., 8, 16, 32, or 64 bits nowadays), software-enabled variable-size virtual structures allow arbitrarily large chunks, and also allow chunks to grow after creation, or to have complex nonlinear topologies. The ability of humans to memorize words, phrases, poems, or formulae of varying length may be based on similar capabilities in brains. Similar considerations apply to the ability to perceive and recognize complex objects with different sorts of structures, such as wheels, cars, lorries, railway trains, etc.

As noted in Minsky (1961), various processes involved in perception, use of language, planning, problem solving, and learning required the ability to create structures that were as large as needed. It also allowed structural units to be complex trees or networks, as opposed to simply being a bit or bit pattern, or a linear array of items.

Support for variable size and variable topology data chunks is not a requirement for *all* kinds of minds. For example, it seems unlikely that insects need it. Even if bees, for instance, are able to learn routes or topographic maps with variable length and topology, this can be implemented in chained associations, which have many of the properties of list struc-

tures mentioned above. (This was discussed in chapter 8 of Sloman 1978, in connection with how children learn about numbers.)

Variable size and variable structure chunking appear to be required for many animal capabilities, including learning about terrain, learning about social relationships, and acquiring new complex behaviors—the kinds of problem solving "tricks" that can be learned by various mammals and birds. It seems to be essential for many aspects of human intelligence, including mathematical problem solving, planning, and linguistic communication (though it has recently become fashionable in some circles to deny this.)

Of course, it is clear that in humans (and other animals) the sizes of the manageable unitary information chunks cannot be *arbitrarily* large, and likewise the data-structures in computers are limited by the addressing mechanism and the physical memory. The infinite tape of a Turing machine is an idealization intended to overcome this, though typically unusable in practice because of performance limitations of a memory that is not randomly addressable. For an organism with a finite life-span operating in real time, however, there is no need to have space to store unbounded structures.

A single animal brain can use different sorts of chunking strategies for different submechanisms. In particular, there are various buffers used for attentive planning or problem solving or reflective thinking, or for real-time processing of sensory data. These all have dedicated mechanisms and limited capacity. So there are limits on the sizes of chunks that can be created and directly manipulated in those short-term memories, some of which (e.g., low level visual arrays) may be much larger than others. Similar size limits need not hold for chunks stored in a longer term memory whose chunks may be too large for instantaneous attention (e.g., a memorized poem, or piano sonata, or route, or one's knowledge of a familiar town or building).

F11. Self-monitoring and self-control

Previously, when discussing sensory transducers in feature F6, I mentioned the possibility of computers having sensory transducers asynchronously detecting and checking *internal* states and processes in addition to *external* ones. The need for this sort of thing has been growing as more and more robust, reliable, secure systems have been developed.

To some extent this self-monitoring can be implemented in software, though it is clumsy and slows things down. For example, it could use the following design:

- Every instruction (in a class of "monitorable instructions") saves a description of what it has just done in some data-structure.
- A process that examines the saved descriptions is time-shared with the other processes. Note that this process may also save descriptions of what it has done.[12]

It may turn out that one of the most important directions for future developments will be to extend these mechanisms. Requirements of advanced AI systems with various kinds of self-awareness may include hardware (firmware?) architectures that provide some kind of general-purpose self-monitoring capability. This might use one or more additional CPUs running the reflective and meta-management processes required to enable such self-percepts to be analyzed, parsed, interpreted, and evaluated, quickly enough to be of use in helping to make the whole system more intelligent.[13]

Simply recording the flow of machine instructions and register contents as a collection of state vectors may not be good enough: it is too low level, though I presume that could be done easily with current technology, at a price. It would be better to have direct support for language-specific or VM-specific records, recording larger chunks. I.e., the self-monitoring mechanisms may have to be parametrizable, like the procedure invocation records in a procedure call stack.

Feature F11 is probably implemented in a very different way in neural systems in brains: e.g., as a neuron fires and sends signals to various other neurons as part of doing its job, there will not be much interference with its performance if an additional connection is made to a monitoring network getting information from many other parts of the system. Compare the "Alarm" mechanisms depicted in our recent papers on the *CogAff* architecture (Sloman 2000a,b, to appear; Sloman and Logan 2000).

The features of modern computing systems listed above can all be seen as continuations of the two trends of development that started in previous centuries and were accelerated by the advent of electronic mechanisms that replaced mechanical and hydraulic storage mechanisms, sensors, switches, and connections. They owe nothing to the idea of a Turing

machine. But they suffice for all the computer-based work that has been done in AI and cognitive science so far, in addition to all the other types of applications of computers.

Even if computers as we know them do not suffice for future developments it seems unlikely that what will be needed is something like a Turing machine. It is more likely that speed, power, and size requirements may have to be met by special-purpose hardware to simulate neural nets, or perhaps chemical computers that perform huge numbers of computations in parallel using interacting molecules such as DNA. Turing machines do not appear to offer anything that is missing.

6 Are There Missing Features?

Some AI researchers may think the above list of features of computers leaves out important features needed for AI, for example, unification, pattern matching, rule-interpreters, and support for logical deduction. However, I believe these additional features are required only for more specialized AI models and applications. Here I have tried to identify what the *most general* features of the physical mechanisms and the virtual machines found in computers are that make them important for AI and cognitive science. This has little or nothing to do with logic, mathematics, or Turing machines, all of which are concerned with rather specialized types of information processing. I shall also try to show, below, that these general features are very similar to features of brain mechanisms, as if evolution discovered the need for these features long before we did.

I have not explicitly included arithmetical capabilities in the eleven main features, though in current computers they are part of the infrastructure for many of the capabilities described above, for example, calculation of offsets in data-structures, handling relative jumps in instruction counters, and scheduling and optimizing processes, etc.

One possible objection to the above list of features is that it omits an important human capability, namely, the ability to cope with infinity. This ability was noted by Kant (1781), who tried to explain our grasp of infinite sets of numbers and infinite space and time in terms of our grasp of a rule that can be applied indefinitely. Frege's (1960) analysis of sentences and their meanings as "compositional" pointed to our ability

to grasp indefinitely complex grammatical and semantic structures. This ability was stressed by Chomsky (1965), who claimed that human beings have a kind of *infinite competence* in their ability to understand or generate sentences of unbounded complexity, for example, arithmetical sentences, and sentences like:

The cat sat on the mat.
The brown cat sat on the mat.
The big brown cat sat on the mat.
The big brown cat sat on the mat in the corner.
. . .

The obvious objection is that human life is finite and the brain is finite, and any individual will produce only a finite number of sentences before dying, and would in any case be unable to understand sentences above a certain length. To this Chomsky replied that even though there are *performance* limitations that get in the way of applying infinite competence, the *competence* exists nevertheless—an answer that left many unconvinced. Questions about the capabilities of computers generate a similar disagreement, as we shall see. One of the important tasks for a theory of mind and computation is to resolve the apparent conflict between infinite competence and bounded physical resources. Both sides are saying something correct, but they talk past each other.

The infinite (or indefinitely extendable) tape of a Turing machine was designed to support this kind of infinite, or unbounded, competence: that was part of Turing's motivation for the unbounded tape, since he was trying to account for human mathematical abilities. Human mathematical abilities appear to be unbounded in various ways. For instance, there is no largest number that we can think of—think of any large number, and we can still think of its square. Likewise, there is no largest algebraic expression whose meaning you can grasp. It seems that for any large expression, you can always append "*+ 1*" and still understand it, at least with a bit of help, for example, by introducing a new name for the complex expression, such as "BigExp," and then thinking about BigExp with "+ 1" appended.

Must this infinite capability be built in to the underlying physical hardware of computers if they are to be able to represent human minds? If so, modern computers would fail, since they have finite memories, and

although it is often possible to add memory to a computer, it will have bounded addressing capabilities that limit the size of addressable memory. For example, a machine with only *N*-bit addressing mechanisms can make use of only 2^N distinct locations.

The situation is even more complex than this suggests, for two reasons. One is that additional addressing capabilities can be built on top of the hardware-addressing mechanisms, as happens when a computer uses a segmented memory, where one address selects a segment and another selects the location in the segment, or when it uses a large backing store, or uses internet addresses to refer to a larger space of possibilities than it can address directly in RAM. However, even those techniques merely *enlarge* the address space, which remains finite.

There is a more subtle point, which does not depend on extending the underlying addressing limits. We know that existing computers, despite their finiteness as *physical* mechanisms, can provide implementations for infinite virtual machines, even though the implementations, if fully tested, turn out to be incomplete. This is because they can store and apply algorithms that have no bounds.

For instance, in many programming languages it is possible to give a recursive definition of the factorial function, which expresses no limit to the size of the input or output numbers. When such a function runs on a computer, either in compiled or interpreted mode, it accurately executes the algorithm for computing the factorial of the given number, up to a point. However, the implementation is usually incomplete in that beyond a certain range of integers the process aborts with some sort of error instead of computing the correct answer.

This does not stop us (correctly) thinking of this as an (albeit only partial) implementation of an infinite virtual machine. That is because every process can be described at different levels of abstraction, and there is a level of abstraction at which the process can be described perfectly correctly as running the recursive algorithm, which does not include any size limits. That is *exactly* what the computer is doing at that level of description, even though *how* it is doing it raises problems if too large a number is given as input.

The point can be expressed by considering which generalizations are true descriptions of the process. For example, the virtual machine that is running on the computer has the feature that if it is given the task of

computing the result of dividing the factorial of any positive number N by the factorial of $N–1$, the result will be N. Similarly, we can say of a sorting algorithm that the result of applying it to any list L will be a new list the same length as L. In both cases the actual physical implementation would break down if the input were too large. If that happened only because there was not enough RAM it might be possible, on some modern computers, to add another bank of memory while the machine was running so as to allow the process to continue; or it might be possible to kill some other less important processes so as to make more memory available. In that case we can see that the memory limit is a *contingent* or *inessential* feature of the implementation.

We can now see that the fact that a process might run out of memory is analogous to the fact that a stray alpha-particle might corrupt the memory, or the power supply might fail, or the building containing it might be struck by a large meteorite, etc. There are vast numbers of possible occurrences that might prevent the normal completion of the calculation, and we don't feel we have to mention all of these when asking what result the computer would produce if given a certain task. There is a perfectly natural way of thinking about processes running on a computer that treats running out of memory as analogous to being bombed. If the building were adequately defended, radar might detect the bomb and divert and destroy it before it hit the building. Likewise, additional mechanisms might protect the process from running out of memory.

It is not easy to extend the addressing limits of a machine at run time. However, it is not beyond the bounds of possibility to have a machine that generalizes what existing paged virtual memory systems do, namely, detect that there is not enough space in the main fast memory, move some unused pages out, and thereafter keep changing the "working set." A more general virtual memory mechanism for a computer that could extend its usable physical store indefinitely would need some addressing scheme that did not use any fixed length structure for specifying locations. In fact, it could in principle use one or more Turing machines with indefinitely extendable tapes. Whatever method is used, the process of specifying the next location to access and the process of fetching or storing something there might get slower and slower, as happens in a Turing machine.

The point of all this is that when something like the standard recursive factorial algorithm runs on a computer the fact that there is a limit to the size of input it can handle is an *inessential* feature of the current implementation, which might be changed while the machine is running. Thus we can ask counterfactual questions about what the result would be if the input were the number 2,000,000 while assuming that implementation obstacles are circumvented as they turn up, including such obstacles as the limited size of the physical universe. Of course, if someone intended the question to be interpreted differently, so that the answer takes account of the current mechanisms in the machine, or the run time modifications that are currently feasible, then this constraint could be explicitly included in the question, and a different answer might then be relevant, for example, "the computer would run out of memory before finding the result." In exactly the same sense, human brains, not least the brains of expert mathematicians, can include implementations of infinite machines, albeit partial and buggy implementations insofar as the machines will falter or fail if given inputs that are too complex, or, for that matter, if tiredness, a distraction, or alcohol or some other drug interferes with performance.

In short, then, although no human mind can actually do anything infinite because of its current implementation, nevertheless humans (and possibly some other animals) have unbounded virtual machines as part of their information-processing architecture. We already know that such virtual machines can be implemented usefully in physical machines that do not support the full theoretical range of functions like factorial, which they partially implement. Thus neither human minds nor robots with humanlike intelligence need have the infinite capacity of a Turing machine in the physical implementation in order to be running a virtual machine with unbounded competence.

What is more, insofar as the human architecture includes a meta-management layer that can (to some extent) inspect the operations of the system, it is able to discover such facts about itself. This is the source of much of the philosophy of mathematics, which is concerned with, among other things, the ability of humans to think about infinite sets. Likewise, a robot whose mind is implemented on a computer and includes a meta-management layer could discover that it had a kind of infinite

competence, and might then become puzzled about how that could be implemented in a finite brain.

All this leaves open the question what precisely is going on when we think about infinite sets such as the set of positive integers, or the transfinite ordinal consisting of the set of all powers of two followed by the set of all powers of three, followed by the set of all powers of five, and so on for all prime numbers. I have discussed this inconclusively in Sloman (2001b) and will not pursue the matter here.

7 Some Implications

The important features of computers listed above (as we know them now, and as they may develop in the near future) have nothing to do with logic, recursive functions, or Turing machines, though systems with the features that are important in computers can (subject to memory limits) model Turing machines and can also be modeled by Turing machines, albeit excessively slowly (assuming that part of a requirement for a functioning mind is to react in time to seize opportunities and avoid dangers).

Turing machines and modern computers can both also model neural nets, chemical soups using DNA molecules to perform computations, and other types of information-processing engines.

Given all this, the view of computers as somehow essentially a form of Turing machine, or as essentially concerned with logic, or recursive functions, is simply mistaken. As indicated above, there is such a mathematical notion of computation, which is the subject of much theoretical computer science, but it is not the primary motivation for the construction or use of computers, nor is it particularly helpful in understanding how computers work or how to use them. A physically implemented Turing machine (with a finite tape) is simply a special case of the general class of computers discussed here, though linking it to sensory or motor transducers might cause problems.

Neither is the idea of a Turing machine relevant to most of the capabilities of human and animal minds, although, as explained above, there is a loose analogy between a Turing machine and some of the formal capabilities of human minds, including the ability to think and reason about infinite structures. This ability must have evolved relatively late and does not play an important role in the vast majority of everyday mental pro-

cesses. It does not appear to be relevant to most animals or to very young children if, as seems likely, these abstract reasoning abilities develop during childhood rather than being there from birth.

8 Two Counterfactual Historical Conjectures

The preceding remarks are closely connected with two conjectures:

1. The development of computers was an inevitable consequence of the availability of electronic technology that provided the potential to overcome the limitations of previously existing machines for controlling other machines, and machines for managing and analyzing information (e.g., commercial databases, airline reservation systems, etc.). Consider what happened during World War II: the need to decipher encrypted messages accelerated the development of electronic calculators. Most of this would have happened anyway, even if Turing and others had not done their metamathematical work on computation, computability, derivability, and completeness.
2. If Turing had never existed, or had never thought of Turing machines, and if computers with the sorts of features listed above had been developed under the pressure of requirements for increasingly sophisticated (and fast) control of physical machines and information processing for practical purposes, then much of AI and cognitive science would have developed exactly as we know them now. AI would not miss the concept of a Turing machine.

Some people interested in logic, theorem proving, etc. might have noticed that computers could be used to implement logic machines of various kinds. But there would have been no notion that computers were somehow *inherently* logical, or *inherently* related to mathematical theorems and their proofs.

These two conjectures follow from the argument that computers were and are, above all, engines for acquiring, storing, analyzing, transforming, and using information, partly to control physical machines and partly to control their own information processing. In this they are just like biological organisms—and that includes using the information both to control complex physical and chemical systems and also to control internal processing within the controller, including processing in virtual machines.

If we think of computers in this way, then it is an empirical question whether a particular physical object does or does not have the kinds of

capabilities listed above. It is not true that every physical object has the collection of features F1 to F11, or even the simpler set F1 to F6. However, finding out which features a very complex machine actually has can be a very difficult reverse-engineering problem. The best available tools at a particular time (e.g., brain scanners) may not be adequate for the job.

Notes

1. Turing machines are often taken to be especially relevant to so-called good old fashioned AI or GOFAI. This term, coined by Haugeland (1985), is used by many people who have read only incomplete and biased accounts of the history of AI, and have no personal experience with working on the problems.

2. Information about looms (including Jacquard looms controlled by punched cards), Hollerith machines used for processing census data, Babbage's and Lovelace's ideas about Babbage's analytical engine, and calculators of various kinds can be found in *Encyclopedia Brittanica*. Internet search engines provide pointers to many more sources. See also Hodges (1992).

3. Pain (2000) describes a particularly interesting analog computer, the "Financephalograph," built by Bill Phillips in 1949. This used hydraulic mechanisms to model economic processes, with considerable success.

4. I conjecture that this mathematical approach to the foundation of mathematics delayed philosophical and psychological understanding of mathematics as it is learned and used by humans. It also has impeded and continues to impede the development of machines that understand numbers as humans do. Our grasp of numbers and operations on numbers is not just a grasp of a collection of formal structures but rather depends on a control architecture capable of performing abstract operations on abstract entities.

5. For more on the parallel between energy and information, see the slides in the following directory: http://www.cs.bham.ac.uk/~axs/misc/talks/.

6. There is a "theological" notion of autonomy, often referred to as "free will," which requires actions to be nonrandom yet not determined by the ballistic or online information available to the agent. This was shown by David Hume to be an incoherent notion.

7. However, many designers of electronic computers did not appreciate the importance of this and separated the memory into code and data, making it difficult to treat program instructions as data, which incremental compilers need to do. This caused problems for a number of AI language developers.

8. It is possible that some formalisms that cannot be manipulated by Turing machines, e.g., formalisms based on continuously varying geometric shapes, will turn out to be relevant to goals of AI, refuting the claimed universality of Turing machines, but that will not be discussed here.

9. I have produced fragments of the following list of features previously, e.g., in chapter 5 of Sloman (1978), but have never really put them all together properly. The list is still provisional, however. I am not aware of any other explicit attempt to assemble such a list, though I believe that all the points are at least intuitively familiar to most practicing AI researchers and software engineers.

10. In Sloman (1985, 1987), I discuss how the system can interpret these sensory changes as due to events in an external environment whose ontology is quite different from the machine's internal ontology.

11. However, the usefulness of general purpose list-processing utilities is usually severely restricted in programming languages that are strongly statically typed, as are most conventional languages.

12. A distributed architecture for self-monitoring using mutual metalevels is outlined in Kennedy (1999).

13. For discussions of the role of a meta-management layer in human minds, see Beaudoin (1994), Wright et al. (1996), Sloman (1997, 1998, 1999, 2000a, 2001a), and Sloman and Logan (2000).

This page intentionally left blank

5

The Practical Logic of Computer Work

Philip E. Agre

Editor's Note

Computationalism, or "Strong AI" as it is often called, is founded on a collection of theories about the mind, in particular a theory of representation holding that knowledge consists in a model of the world, a theory of perception attempting to build a mental model by working backward from sense-impressions, and a theory of action viewing behavior as resulting from the execution of plans. Phil Agre isolates five conceptual tensions inherent in this collection of theories, which underwrite AI's history (and with it the history of computationalism) and determine its practical logic of research: the oppositions between mind and world, mind and body, mental activity and perception, plans and behavior, and finally abstract ideals and concrete things. Having accompanied AI from its very beginnings, these five tensions are deeply yet tacitly embedded in the intellectual discourse of the field (i.e., in its language and methods), and often surface to cause a bewildering array of difficulties in computational practice. Although various strategies to cope with these tensions are discussed, elaborated, accepted, or rejected in written expositions, Agre points out, the tensions themselves are not recognized, reflected on, or accepted as part of Western intellectual history, but put aside and compensated for, often by a formalist attempt to cleanse language and liberate AI of all its cultural and contextual underpinnings. Not surprisingly, these undigested tensions are still very much present and reveal themselves in a recurrent pattern that Agre calls "dissociation": the two concepts underwriting each opposition are overtly recognized as distinct by the community, but are then covertly (and unintentionally) conflated in particular writings. While the pattern of coping with these tensions is the same, instances of dissociations may take shape in various ways, as exemplified by Agre for each of the five oppositions. The dissociation between mind and body, for example, is apparent from the typical division of labor in "planning" in AI, where "the mind" generates the plan, and "the body" executes it. The dissociation between mind and world, on the other hand, results from the distinction between the inside and the outside of an agent, and eventually transforms into a conflation of the two by employing the very notion of "world model" for the idealized, complete representation of the world within the agent. Yet another dissociation, that between plans and behavior, becomes eventually conflated by using the term "plan" to refer to any

structure in behavior (e.g., a sequence of actions). In the end, Agre blames these unresolved tensions on a certain kind of transcendentalism in AI, which attempts to "hold something apart from and above material reality." Rather than trying to view the world as negative and chaotic and the mind as positive and in charge of ordering the worldly chaos, he concludes that AI needs to recognize the order inherent in the world and thus advocates a "critical" technical practice that may be able to listen to and learn from reality by building systems and understanding the ways in which they do and do not work.

1 Inscription Errors

In his reading of John Dee's sixteenth-century commentary on Euclid, Knoespel (1987) describes what he calls the "narrative matter of mathematics": the discursive work by which an artifact such as a drawing on paper is exhibited as embodying a mathematical structure such as a Euclidean triangle. This process, which is familiar enough to modern technologists as the work of mathematical formalization, consists of a transformation of language into something quite different from language, namely mathematics, and its challenge is to circumvent or erase those aspects of language that are incompatible with the claim that mathematics represents the world in a transparent, unmediated, ahistorical way. Knoespel explains the larger significance of the point like this:

> . . . the reification of geometry in architecture and technology has enormous implications for language. Once geometry becomes manifest in artifacts, these artifacts retain an authority radically different from that accessible to natural language. By virtue of their being manifested as physical objects they acquire what appears as an autonomy utterly separated from language. The apparent separation of both architecture and technology from language has great significance, for it works to repress the linguistic framework that has allowed them to come into being. (1987: 42)

Brian Smith (1996) makes a similar point when he speaks of "inscription errors": inscribing one's discourse into an artifact and then turning around and "discovering" it there. The hidden connections between artifacts and language are particularly significant in the case of another large category of artifacts, namely computers. The design of a computer begins with formalization—an intensive and highly skilled type of work on language. A computer is, in an important sense, a machine that we build so that we can talk about it; it is successful if its operation can be narrated using a certain vocabulary. As the machine is set to running, we too easily

overlook the complicated reflexive relationship between the designer and user on which the narratability of the machine's operation depends (see Suchman and Trigg 1993). The corpus of language that was transformed in producing the machine, like any discourse or text, is necessarily embedded in an institutional and historical context, and the machine itself must therefore be understood as being, in some sense, a text (see Woolgar 1991).

For practitioners of a computational research tradition such as artificial intelligence (AI), however, the textuality of computers is a distraction at best, to the extent that it is even comprehensible as an analysis of technical work. Like John Dee, the practitioners of AI with whom I worked for several years, and among whom I was trained, regarded formalization very differently, as a means precisely of freeing their work from the unruliness and imprecision of vernacular language. They believed that, by defining their vocabulary in rigorous mathematical terms, they could leave behind the network of assumptions and associations that might have attached to their words through the sedimentation of intellectual history. As a result, I want to argue, the field of AI has been left with no effective way of recognizing the systemic difficulties that have arisen as the unfinished business of history has irrupted in the middle of an intendedly ahistorical technical practice. My purpose in diagnosing this situation is not to discredit mathematics, technology, or even AI. Instead, I want to envision a critical technical practice: a technical practice within which such reflection on language and history, ideas and institutions, is part and parcel of technical work itself (Agre 1995, 1997b).

2 Patterns of Dissociation

My point of departure is what AI people themselves have taken to calling Good Old-Fashioned AI (GOFAI)—the complex of metaphors and techniques from which the great majority of the AI literature until recently has begun. Even those projects that do not fully embrace this standard theory are usually well understood as incrementally relaxing its assumptions in an attempt to resolve its seemingly inherent tensions.

The standard theory encompasses, among other things, particular theories of representation, perception, and action:

• The theory of representation holds that knowledge consists in a model of the world, formalized (when it *is* formalized) in terms of the Platonic analysis of meaning in the tradition of Frege and Tarski. Individual things in the world (people, robots, places, tools, blocks, and so on) are represented on a one-for-one basis by constant symbols in a formal or informal logic, and regularities, conventions, and logical connections in the world are represented by quantified propositions.

• Perception is a kind of inverse optics (Hurlbert and Poggio 1988): building a mental model of the world by working backward from sense-impressions, inferring what in the world might have produced them. The computational neurophysiologist David Marr (1982) gave methodological form to this metaphor with his suggestion that the visual cortex consists of a set of modules that reconstruct a detailed three-dimensional model of the world by reasoning backward from sense impressions to the processes in the world that must have produced them.

• Action, finally, is conducted through the execution of mental constructs called plans, understood roughly as computer programs. Plans are constructed by searching through a space of potential future sequences of events, using one's world models to simulate the consequences of possible actions.

This theory is extraordinarily productive, in the sense that it provides a plausible starting point for research and model-building on nearly any topic of human or machine life. It is accordingly the basis of a vast literature. It is founded in several oppositions, and I want to focus on the specific way in which these oppositions figure in the history of the field. They are as follows:

1. mind versus world
2. mental activity versus perception
3. plans versus behavior
4. the mind versus the body
5. abstract ideals versus concrete things

Throughout AI, one encounters a certain pattern in the handling of these oppositions. I will refer to this pattern as "dissociation." Dissociation has two moments: an overt distinction between the two terms and a covert conflation of them. As the language of overt and covert may suggest, dissociation is not consciously avowed. It must be sought in the internal tensions of texts and in the dynamics of technical work. Dissociation occurs in many ways, none of which is fixed or stable. To the con-

trary, much of the history of AI can be understood as attempts to comprehend and manage the signs of trouble that originate in dissociation and that AI practitioners can recognize within the conceptual system of the field. One purpose of a critical technical practice is to recognize such troubles more fully and to diagnose them more deeply.

It would be impossible to demonstrate in the entire literature of AI the workings of the five dissociations that I enumerated above. Each AI text needs to be treated on its own terms, and no strong generalizations are possible. Instead, I will review the workings of the five dissociations as they become manifest in particular texts and projects. It would require a separate and much larger effort to trace the evolution of the various dissociations through the historical development of the literature.

3 Mind versus World

The distinction between mind and world has obviously been construed in many ways by different philosophical systems, but in AI it has been operationalized in terms of a sharp qualitative distinction between the inside of the machine and the world outside. This image of the mind as an internal space, what Lakoff and Johnson (1980) call the "container" metaphor, underlies a large proportion of the discourse of AI (Agre 1997a). The difficulty is that it is hard to maintain accurate knowledge of a world that is understood to be so distinct (cf. Bordo 1987). Viewed at such a distance, the world seems profoundly uncertain, and the likelihood that other people and things will change the world according to their own agendas makes ordinary life seem nearly impossible.

As a result, it is a common idealization in the field to suppose that one's world model is complete in every relevant respect and stays up-to-date automatically. Mind and world, having been distinguished at the outset, are covertly conflated in the very concept of a world model—a simulacrum of the world in one's head. This conflation of mind with world is facilitated by the use of research "domains" such as geometry that are readily understood in mathematical terms (cf. Walkerdine 1988). This practice first took hold with the work on logical theorem proving by Newell, Shaw, and Simon (1960) and on chess by Newell and Simon (1963). Even in domains that involve physical objects, it is common for

AI people (and computer scientists in general) to employ the same words to name both the representations in a machine and the things that those representations represent. The erasure of language that Knoespel discovered in Dee is here employed on a grand scale to facilitate the conflation, not just of mathematics and the physical world, but of representation and reality in general.

The point is not that AI researchers do this deliberately. Quite the contrary, their work is well understood as an attempt *not* to do it. Having acquired the conflation of mind and world through the discursive practices they inherited from earlier intellectual projects, AI researchers continually find the conflation reinforced through the practical logic of their research. If I may risk a mathematical metaphor, the dissociative relationship of mind and world is an attractor: although unsatisfactory as a theory of human life, the dissociation is also difficult to escape by incremental modifications because the covert conflation of mind and world solves, even if pathologically, a problem for which no less pathological solution is known. The mathematical metaphor here is misleading, though, in that the terrain is a function of consciousness: which "moves" in the space of research proposals seem comprehensible and well motivated is itself a function of the critical awareness through which the symptoms of technical trouble are perceived and interpreted. The point, therefore, is not that AI researchers believe that the mind and world are the same, or that they are incapable of grasping the idea that they are different. Rather, practices that tend to conflate the two are dispersed throughout the discourse and practices of the field; researchers are socialized into the discourse and practices through their training in the field, and the field does not provide the critical tools that would be necessary to detect this pattern and reverse it. So long as the larger pattern remains ungrasped, the attractor retains its grip on the research.

4 Mental Activity versus Perception

AI originated as part of the cognitivist movement whose other was behaviorism. In combating the behaviorist hegemony, the task of cognitivism was to present itself as scientific by the same standard as behaviorism did, while also opposing itself as squarely as possible to behaviorism on a substantive level. As Karl Lashley noted in an influential early symposium

paper (1951), the behaviorist theory of action was stimulus-chaining: the idea that each action in a serially ordered sequence arises as a response to, among other things, the effects of the previous action in the sequence. Lashley, however, observed that the elementary actions involved in speech—individual phonemes and words—have no necessary relation to one another. The same phoneme, for example, might be found adjacent to a large number of other phonemes in the context of particular words (1951: 115–116). It is therefore impossible that stimulus-response connections among particular linguistic elements could explain the ways in which they are arranged in any particular utterance. He therefore urged that serially ordered action must be understood in terms of an internal organizing principle.

In his effort to oppose himself to the behaviorist story, Lashley equivocates on the role of perception in organizing behavior. A fair-minded reader will reconstruct his argument as advocating a two-factor theory of behavior: continual perceptual inputs into the ongoing mental process (1951: 112, 131), and continual outputs from that process in the form of spoken phonemes and other actions drawn from a language-like repertoire. But in fact Lashley provides little substantive account of the role of perception, emphasizing instead that serial behavior is planned and organized in advance and performed in scripted wholes. What is more, his use of language as a paradigm case of action leaves little room for perceptual input. He imagines linguistic utterances to arise through the interaction of three factors: the "expressive elements," including phonemes and words, a "determining tendency" that encodes the information to be conveyed without itself having any internal structure, and the "syntax of the act," which he defines as "an habitual order or mode of relating the expressive elements; a generalized pattern or schema of integration which may be imposed upon a wide range and a wide variety of specific acts" (1951: 122).

Thus begins the practice in cognitive science of using language, and specifically the production of grammatical utterances, as the prototypical case of human action. Note, however, that the resulting pattern of behavior results solely from the idea one wishes one's actions to "express," and not from any stimulus in the world. This restriction can go unnoticed in the case of language because phonemes and words follow one another so rapidly, and their formal relationships are so densely organized, that

the outside world has little time to influence their ordering. Of course, the ongoing interaction between speakers and hearers *can* influence an utterance as it unfolds (Goodwin 1981), but this idea plays no role in Lashley's argument, nor indeed in the mainstream of cognitive science.

Dissociation in Lashley takes a distinct form: rather than uniting bodily through systematic ambiguity, the two terms are brought together in two separate theories that are then superimposed. One of these theories envisions action arising through a formal order being imposed on a population of expressive elements; this theory of the mental scripting of action is given the more overt billing. The other theory imagines the mind to be a holistic process—"a great network of reverberatory circuits, constantly active" (1951: 131)—into which perception is simply one more influence among many. The two theories are attached at various points, but not so closely that the impossibility of perception's role ever becomes obtrusive.

5 Plans versus Behavior

Although rarely cited in the modern literature, the *locus classicus* for AI discussion of plans is Miller, Galanter, and Pribram's *Plans and the Structure of Behavior* (1960). Miller, Galanter, and Pribram's starting point, like Lashley's, was the intellectual dominance of behaviorism. Their organizing question, therefore, is whether something must be imagined to mediate between the observable categories of stimulus and response. They find half of their answer in Boulding's (1956) notion of "the Image"—a vague precursor to the concept of a world model. They observe that the Image explains knowledge and not action, and they ask why behavior has any structure. The reason, they suggest (1960: 16), is because behavior arises from a mental entity that has that same structure, namely a Plan. (They capitalized the word.) They, like Lashley, would seem to have shifted from a theory whose only explanatory factor is perception to a theory in which perception, and thus the outside world as a whole, plays no role at all.

Although it is openly speculative and presents no computer demonstrations or laboratory experiments of its own, *Plans and the Structure of Behavior* served as a rhetoric for the emerging cognitivist movement. Its most distinctive rhetorical device was a systematic conflation of behavior and Plans. To this day, it is customary in AI research to refer to any

structure in behavior as a plan (lower case now, but with the same meaning), not as a hypothesis but as a simple matter of terminology. Thus, for example, Cohen et al. (1989; cited by Vera and Simon 1993: 26), echoing the earliest arguments of Lashley and Miller, Galanter, and Pribram, ascribe to my own work (Agre and Chapman 1987) the idea that "longer-term plans can emerge from compositions of reflexes." The very suggestion is not coherent unless "plans" (in the mind) are confused with "sequences of action" (in the world).

In fact, Miller, Galanter, and Pribram took Lashley's dissociation between perception and thought a step further. Close reading demonstrates the presence of two distinct theories: one speaking of discrete, prescripted Plans and another speaking of a single Plan that unfolds incrementally through time. Thus, for example, having defined their concept of a Plan as something like a computer program that controls behavior, they casually amend their definition by saying: "Moreover, we shall also use the term 'Plan' to designate a rough sketch of some course of action, just the major topic headings in the outline, as well as the completely detailed specification of every detailed operation" (1960: 17). The former theory, the one that describes a repertoire of discrete Plans, is the official theory: it is the theory that explains the structure of behavior. Yet it is an unsatisfactory theory in many ways. Computer programs, for example, do need to be spelled out far in advance, and they must anticipate every possible contingency of their execution; and so the official theory is overlaid with the unofficial theory of a single Plan that can be constructed in a more improvisational manner as an ongoing response to perception. The authors return to the matter again by offering a theory of Plans based on then-current ideas of servocontrol (1960: 26–39). This third theory—the so-called TOTE units (test, operate, test, exit) that supposedly provide a "unit of analysis" for Plans—is laid atop the earlier two with little overt sense of their incompatibility; in any case, it is the equivocal relation between the first two theories that set the agenda for subsequent research.

Most of this subsequent research focused exclusively on the construction of discrete plans (Allen, Hendler, and Tate 1990; Georgeff 1987), thus obviating the need for a theory of improvisation. It is, however, computationally very difficult to construct a Plan that will work reliably in a world of any great complexity. The problem is not precisely the size of the space of possible plans, since subtle use of probabilistic search

techniques can overwhelm even quite large planning search problems (Gomes, Selman, and Kautz 1998). Rather, in a dynamic world it is essentially impossible to know everything that would be needed to conduct such a search. As a result, the second of Miller, Galanter, and Pribram's theories—the incremental, on-the-fly construction of a partial Plan—resurfaced in the form of "reactive planning" in the 1980s (e.g., Firby 1987; Fox and Smith 1984). One term of the dissociation, having been suppressed, returned. Yet when it did so, its internal relationship to the dominant theory was hardly recognized, and research in that era saw several attempts to resynthesize the two approaches (e.g., Payton, Rosenblatt, and Keirsey 1990), reproducing in the architecture of cognitive models the conflation that occurred more covertly in Miller, Galanter, and Pribram's text.

6 The Mind versus the Body

The AI literature often dissociates mind and body, and here again the covert conflation between the two terms takes a different form. It can be seen in the first computer implementation of the plans-and-execution theory, the STRIPS model of Fikes, Hart, and Nilsson (1972), which automatically constructed plans and then executed them by directing the movements of a robot. These authors' dilemma is that it is hard to predict in enough detail how the world will unfold as the robot executes its plans, and yet it is so expensive to construct a new plan that it seems necessary to execute as much of each plan as possible. At one point they suggest simply executing the first step of each new plan and then constructing a new plan from scratch, but they reject the idea as impractical.

> One of the novel elements introduced into artificial intelligence research by work on robots is the study of execution strategies and how they interact with planning activities. Since robot plans must ultimately be executed in the real world by a mechanical device, as opposed to being carried out in a mathematical space or by a simulator, consideration must be given by the executor to the possibility that operations in the plan may not accomplish what they were intended to, that data obtained from sensory devices may be inaccurate, and that mechanical tolerances may introduce errors as the plan is executed.
>
> *Many of these problems of plan execution would disappear if our system generated a whole new plan after each execution step. Obviously, such a strategy would be too costly,* so we instead seek a plan execution scheme with the following properties:

1. When new information obtained during plan execution implies that some remaining portion of the plan need not be executed, the executor should recognize such information and omit the unneeded plan steps.
2. When execution of some portion of the plan fails to achieve the intended results, the executor should recognize the failure and either direct reexecution of some portion of the plan or, as a default, call for a replanning activity. (Fikes, Hart, and Nilsson 1972: 268; emphasis added)

The dissociation between the mind, which constructs plans, and the body, which executes them, manifests itself here in a seeming attempt by planning and execution to become reunited into a single compound process, or at least to become intertwined with one another in rapid alternation. The impracticality of the dissociation is revealed through the process of system-building.

It is here, in the design and construction of systems, that AI learns most of its lessons, and not in formal experimentation with the systems once they are running. If I seem to anthropomorphize the desire of mind and body to reverse their dissociation, this is only to give voice to a common experience of system designers: the tangible pressures that seem to push the design process in one direction or another.

7 Abstract Ideals versus Concrete Things

The final dissociation is between abstraction and concrete reality in the tradition of Fregean semantics. Historically, theories of semantics have drifted between two poles: psychologism, the theory that meanings are mental entities, and Platonism, the theory that meanings reside in a timeless, extensionless realm of ideals. Yet this distinction has made remarkably little difference in practice. The mind, as it has been constructed through the system of dissociations that I have already described, resembles the realm of Platonic ideals in many ways. The use of mathematical domains helps to blur the distinction between Platonic ideals and real human ideas, inasmuch as mathematical entities resemble Platonic ideals and are now routinely employed to formalize Platonic theories of the semantics of both language and logic.

The problem arises whenever the mind must be understood as participating in the historical, causal world. The problem manifests itself in many ways, but it can be clearly grasped already in Frege's own theory. Frege's theory of sense, that is, the cognitive content of a representation,

conflates incompatible goals. The sense of a representation is supposed to be a Platonic ideal, independent of place and time, and yet it is also held to incorporate specific objects such as a car or a person. The problem is particularly acute in the case of indexical representations, whose sense is plainly dependent on the concrete historical circumstances of their use. I can use words like "here" and "now" perfectly well without knowing where I am or what time it is, and yet the sense of my utterance involves those specific places and times. The trouble arises because Frege's theory of sense, and a whole subsequent tradition of semantic theories, attempts to capture the content of thoughts and utterances without reference to the embodied activities and relationships with which they are used. Burge (1979: 426) summarizes the problem:

> Frege appears to be caught between two objectives that he had for his notion of sense. He wanted the notion to function as conceptual representation for a thinker (though in principle accessible to various thinkers, except in special cases). And he wanted it uniquely to determine the referents of linguistic expressions and mental or linguistic acts. The problem is that people have thoughts about individuals that they do not individuate conceptually.

As the realm of Platonic abstractions is conflated with the realm of concrete activity, the semantic formalism takes on a covertly material character and the theory of mind takes on a covertly disembodied character.

8 Conclusion

These, then, are five dissociations that have organized the practical logic of research within the leading tradition of artificial intelligence. The interaction among these dissociations defines a complex and variegated field whose contradictory pressures must be reconciled somehow in each project in its own unique way. In most cases I have chosen particular projects to illustrate the practical logic of dissociation. Other projects may well be discovered to manage the tensions in a different way. Yet few AI projects, if any, have reflected a systematic understanding of them, much less any resolution.

Where do these dissociations originate? Do all conceptual distinctions lead to dissociations when they are embodied in technical work? When AI work is framed in terms of these dissociations, it becomes obvious

that AI is part of the history of Western thought. It has inherited certain discourses from that history about matters such as mind and world, and it has inscribed those discourses in computing machinery. The whole point of this kind of technical model-building is conceptual clarification and empirical evaluation, and yet AI has failed either to clarify or to evaluate the concepts it has inherited. Quite the contrary, by attempting to transcend the historicity of its inherited language, it has blunted its own awareness of the internal tensions that this language contains. The tensions have gone underground, emerging through substantive assumptions, linguistic ambiguities, theoretical equivocations, technical impasses, and ontological confusions.

Yet the project of artificial intelligence has not been a failure. The discipline of technical implementation, considered in another way, really has done its job. The dissociations of AI have refused to go away, and in the practical work of model-building they have ceaselessly reasserted themselves. As AI practitioners have tried to manage all of the many cross-cutting tensions in their work, they have made the generative principle of those tensions ever more visible to those with the critical means to look. This is the motivation for my advocacy of a critical technical practice and the basis of my positive hopes for it. Bad theories of human existence, we might conjecture, do not simply fail to correspond to the data; they are in fact impracticable, in that nobody and nothing could live in the ways they describe. As AI people learn to expand their understanding of the ways in which a system can "work" or "not work," AI can perhaps become a means of listening to reality and of learning from it. The key to this transition is a reversal of the ideological function of technology that Knoespel described at the outset. The texts that we inscribe in computing machinery are texts in the fullest sense. In particular, they always have greater depths than we are aware of, and they always encode sophisticated strategies for managing the contradictory experience of the world that we have inherited from the past. Computers and formalization do not eliminate all this; quite the contrary, they make it more visible.

And what exactly has AI made visible in this fashion? Running through all of the dissociations I mentioned is a recurring theme, a kind of transcendentalism that attempts to hold something apart from and above material reality. The transcendentalist philosophy of AI is that the outside

world is, to use the military language that AI took over during the 1980s (Hendler 1990), uncertain, unpredictable, and changing. The dissociations portray the world as a negative, chaotic principle and the mind as a positive, ordering principle. What we might tentatively conclude, in the irreducibly intuitive and practical way in which such conclusions might possibly be drawn from technical work, is that it doesn't work that way. If not for the positive ordering principles inherent in the world itself, life would be impossible. Man, in this traditional sense, can live not by imposing order, but only by participating in it. Only from this standpoint amid the order of the world does the world itself become visible; only then does the question usefully arise of how best to live in the world, and how best to change it.

6

Symbol Grounding and the Origin of Language

Stevan Harnad

Editor's Note

If computationalism is right and thinking is the computational manipulation of symbols on the basis of their formal properties, from where do these symbols derive their meaning? This question leads to what Stevan Harnad has called the *symbol grounding problem*. In Harnad's view, computationalist theories of meaning cannot provide a satisfactory answer: internal symbols of a computational system, despite being systematically interpretable, simply do not "contain" their meanings, which rather reside in the heads of those who use computations as tools. So how can thoughts be about something—how can words denote anything at all? One possibility is to extend the boundaries of the symbol system to include the environment that contains the referents of the symbols to make sense out of the "aboutness" relation. According to Harnad, there is no need to resort to such "wide notions of meaning," as it would essentially entail an extension of psychology beyond the boundaries of the contents of the brain, which is independently problematic. Rather, a "narrow" account can be successful if symbols are not taken to be connected to distal objects in the environment, but rather to the "proximal shadows that the distal objects cast on the system's sensorimotor surfaces." In other words, symbols that denote categories of objects (such as "mushroom") must be grounded in the capacity to sort, label, and interact with the proximal sensorimotor projections of their distal category-members in a way that coheres systematically with their semantic interpretations, both for individual symbols, and for symbols strung together to express truth-value-bearing propositions. Nevertheless, not all categories need to be grounded this way. This is where language enters the picture and allows us to "steal" categories quickly and effortlessly through hearsay instead of having to earn them the hard way, through risky and time-consuming sensorimotor trial-and-error learning, guided by corrective feedback from the consequences of miscategorization. Linguistic "theft," however, is not possible unless some of the denoting symbols of language are previously grounded directly in categories that have been earned through sensorimotor toil or else in categories that have been inherited through Darwinian theft (i.e., have been acquired genetically as a consequence of evolutionary trial and error plus the sensorimotor toil of evolutionary ancestors). Language also enables us to learn the meaning of an expression that we could not have acquired through

sensorimotor projections, either because they are identical for different distal objects, or because there is no object that could have a projection, as Harnad illustrates with the imaginative "Peekaboo Unicorn." He concludes that the ontological status of the referent of a term is irrelevant to the cognitive issue of its meaning.

1 Meaning: Narrow and Wide

Philosophers tell us that there are two approaches to explaining what our thoughts are about, one wide and one narrow. According to the wide approach, the object I am thinking about, that physical thing out there in the world, is to be reckoned as part of the meaning of that thought of mine about it; meaning is wider than my head. According to the narrow approach, the locus of any meaning of my thoughts is inside my head; meaning can be no bigger than my head. The question is: should psychologists who aspire to study and explain the meaning of thoughts adopt a wide or a narrow approach?

Here is an advantage of a wide approach: as wide meaning encompasses both the internal thought and its external object, a complete explanation of wide meaning would leave nothing out. Once you have arrived at a successful explanation, there are no further awkward questions to be asked about how thoughts get "connected" to what they mean, because what they mean is in a sense already part of what thoughts are. But there are disadvantages for a psychologist adopting this approach, because it would require him to be so much more than a psychologist: he would have to be an authority not only on what goes on in the head, but also on what goes on in the world, in order to be able to cover all the territory over which thoughts can range.

We can illustrate the plight of the wide psychological theorist with an analogy to the roboticist: if one were trying to do "wide robotics," accounting not only for everything that goes on inside the robot, but also for what goes on in the world, one would have to model both the robot and the world. One would have to design a virtual world and then describe the robot's states as encompassing both its internal states and the states of the virtual world in which it was situated. In this ecumenical era of "situated" robotics (Hallam and Malcolm 1994) this might not sound like such a bad thing, but in fact wide robotics would be quite contrary to the spirit and method of situated robotics, which emphasizes

using the real world to test and guide the design of the robot precisely because it is too hard to second-guess the world in any virtual-world model of it ("the world is its own best model"). At some point, the "frame problem" (Pylyshyn 1987; Harnad 1993b) always arises; this is where one has failed to second-guess enough about the real world in designing the robot's virtual world, and hence a robot that functions perfectly well in the virtual world proves to be helpless in the real world.

So perhaps it's better to model only the robot, and leave the modeling of the real world to cosmologists. The counterpart of this moral for the psychologist would be that he should restrict himself to the narrow domain of what's going on in the head, and likewise leave the wide world to the cosmologists.

2 The Symbol Grounding Problem

But there are disadvantages to the narrow approach too, for, having explained the narrow state of affairs in the head, there is the problem of ensuring that these states connect with the wide world in the right way. One prominent failure in this regard is the symbolic model of the mind ("computationalism"), according to which cognition is computation: thinking is symbol manipulation (Pylyshyn 1984). After the initial and promising successes of artificial intelligence, and emboldened by the virtually universal power that was ascribed to computation by the Church-Turing Thesis (according to which just about everything in the world can be simulated computationally), computationalism seems to have run aground precisely on the problem of meaning: the symbol-grounding problem (Harnad 1990, 2001b). For in a narrow computational or "language of thought" theory of meaning, thoughts are just strings of symbols, and thinking is merely the rule-governed manipulation of those symbols, as in a computer. But the symbols in such a symbol system, although they are systematically interpretable as meaning what they mean, nevertheless fail to "contain" their meanings; instead, the meanings reside in the heads of outside interpreters who use the computations as tools. As such, symbol systems are not viable candidates for what is going on in the heads of those outside interpreters, on pain of infinite regress.

So cognition must be something more than computation (Harnad 1994). Was the fatal failing of computation that it was a narrow theory,

operating only on the arbitrary shapes of its internal symbols and leaving out their external meaning? Do we have no choice but to subsume the wide world after all?

Subsuming the wide world has not been the response of the cognitive modeling community. Some have chosen to abandon symbols and their attendant grounding problems altogether (Brooks 1991), and turn instead to other kinds of internal mechanisms, nonsymbolic ones, such as neural nets (Damper and Harnad 2000) or other dynamical systems (van Gelder 1998). Others have held onto symbol systems and emphasized that the solution to the grounding problem is to connect symbol systems to the world in the right way (Fodor 1987, 1994). The problem with this approach is that it provides no clue as to how one is to go about connecting symbol systems to the world in the right way, the way that links symbols to their meanings. Hence one does have reason to suspect that this strategy is rather like saying that the problem of a robot that has succeeded in its virtual world but failed in the real world is that it has not been connected with the real world in the right way: the notion of the "right connection" may conceal a multitude of sins of omission that in the end amount to a failure of the model rather than the connection. (Or, to put it another way, the hardest problems of cognitive modeling may be in designing the robot's internal states in such a way that they do manage to connect to the world in the "right way.")

But let us grant that if the symbolic approach ever succeeds in connecting its meaningless symbols to the world in the right way, this will amount to a kind of wide theory of meaning, encompassing the internal symbols and their external meanings via the yet-to-be-announced "causal connection." Is there a narrow approach that holds onto symbols rather than giving up on them altogether, but attempts to ground them on the basis of purely *internal* resources?

There is a hybrid approach that in a sense internalizes the problem of finding the connection between symbols and their meanings; but instead of looking for a connection between symbols and the wide world, it looks only for a connection between symbols and the sensorimotor projections of the *kinds* of things the symbols designate: it is not a connection between symbols and the distal objects they designate but a connection between symbols and the proximal "shadows" that the distal objects cast on the system's sensorimotor surfaces (Harnad 1987; Harnad et al. 1991).

3 Categorization: A Robotic Capacity

Such hybrid systems place great emphasis on one particular capacity that we all have, and that is the capacity to categorize, to sort the blooming, buzzing confusion that reaches our sensorimotor surfaces into the relatively orderly taxonomic kinds marked out by our differential *responses* to it—including everything from instrumental responses such as eating, fleeing from, manipulating or mating with some kinds of things and not others, to assigning a unique, arbitrary name to some kinds of things and not others (Harnad 1987).

It is easy to forget that our categorization capacity is indeed a *sensorimotor* capacity. In the case of instrumental responses, based on the Gibsonian invariants "afforded" by our sensorimotor interactions with the world, what we tend to forget is that these nonarbitrary but differential responses are actually acts of categorization too, partitioning inputs into those you do this with and those you do that with. And, in the case of unambiguously categorical acts of naming, what we forget is that this is a sensorimotor transaction too, albeit one subtended by an arbitrary response (a symbol) rather than a nonarbitrary one.

So the capacity to sort our sensorimotor projections into categories on the basis of sensorimotor interactions with the distal objects of which they are the proximal projections is undeniably a sensorimotor capacity; indeed, we might just as well call it a robotic capacity. The narrow hybrid approach to symbol grounding to which I referred attempts to connect the proximal sensorimotor projections of distal objects and events to either the instrumental responses or the arbitrary names that successfully sort them according to what is adaptive for the hybrid system. The instrumental part is just an adaptive robot; but the arbitrary category names (symbols) open up for the system a new world of possibilities whose virtue is best described as the advantage of theft over honest toil (Cangelosi and Harnad 2000).

4 Acquiring Categories by Sensorimotor Toil and Darwinian Theft

Before defining sensorimotor toil versus symbolic theft, let me define a more primitive form of theft, which I will call *Darwinian theft*. In each case, what is being either stolen or earned by honest toil are sensorimotor *categories*. It is undeniable that we have the capacity to detect the proxi-

mal sensorimotor projections of some distal categories without ever having to "earn" that capability in any way, because we are born that way. Just as the frog is born with its Darwinian legacy of bug-detectors, we also arrive with a "prepared" repertoire of invariance-detectors that pick out certain salient shapes or sounds from the otherwise blooming, buzzing confusion reaching our sensorimotor surfaces without our first having to learn them by trial and error: for sensorimotor trial and error, guided by corrective feedback from the consequences of miscategorizing things (Skinnerian learning), is what I am calling "honest toil."[1]

We must infer that our brains, in addition to whatever prepared category-detectors they may be born with, are also born with the capacity to learn to distinguish the sensorimotor projections of members from those of nonmembers of countless categories through supervised learning (Andrews et al. 1998; Pevtzow and Harnad 1997), that is, through trial and error, guided by corrective feedback from the consequences of categorizing correctly and incorrectly (Harnad 1996a). The exact mechanism which learns the invariants in the sensorimotor projection that eventually allow the system to sort correctly is not known, but neural nets are a natural candidate: learning to categorize by honest toil is what nets seem to do best (Harnad 1993a).[2]

Does a system such as the one described so far—one that is able to sort its proximal projections, partly by Darwinian theft, partly by sensorimotor toil—suggest a narrow or a wide explanation of meaning (insofar as it suggests anything about meaning at all)? It seems clear that whatever distal objects such a system may be picking out, the system is working entirely on the basis of their proximal projections, and whatever connection those may have with their respective distal objects is entirely external to the system; indeed, the system would have no way of distinguishing distal objects if the distinction were not somehow preserved in their proximal projections—at least, no way without the assistance either of sensorimotor aids of some kind, or of another form of theft, a much more powerful one, to which I will return shortly.

5 Categories Are Context-Dependent, Provisional, and Approximate

So for the time being we note that the categories of such a Darwinian/Skinnerian system are only approximate, provisional, context-dependent ones (Harnad 1987). They depend on the sample of proximal projections

of distal objects the system happens to have encountered during its Skinnerian history of sensorimotor toil (or what its ancestors happen to have encountered, as reflected by the detectors prepared by Darwinian theft): whatever systematic relation there might be between them and their distal objects is external and inaccessible to the system.

Philosophers tend, at this juncture, to raise the so-called problem of error: if the system interacts only with its own proximal projections, how can its categories be said to be right or wrong? The answer is inherent in the very definition of sensorimotor toil: the categories are learned on the basis of corrective feedback from the consequences of *mis*categorization. It is an error to eat a toadstool, because it makes you sick. But whether the proximal projection of what *looks* like a safely edible mushroom is really an edible mushroom, or a rare form of toadstool with an indistinguishable proximal projection that does not make you sick but produces birth defects in your great-grandchildren, is something of which a narrow system of the kind described so far must be conceded to be irremediably ignorant. What is the true, wide extension of its mushroom category, then? Does its membership include only edible mushrooms that produce no immediate or tardive negative effects? Or does it include edible mushrooms plus genetically damaging toadstools? This is a distal difference that makes no detectable proximal difference to a narrow system such as this one. Does it invalidate a proximal approach to meaning?

Let's ask ourselves why we might think it might invalidate proximal meaning. First, *we* know that two different kinds of distal objects can produce the same proximal projections for a system such as this one. So we know that its narrow detector has not captured this difference. Hence, by analogy with the argument we made earlier against ungrounded symbol systems, such a narrow system is not a viable candidate for what is going on in our heads, because we ourselves are clearly capable of making the distal distinction that such a narrow system could not make.

6 Distinguishing the Proximally Indistinguishable

But in what sense are we able to make that distal distinction? I can think of three senses in which we can. One of them is based, trivially, on sensorimotor prostheses: an instrument could detect whether the mushroom was tardively carcinogenic, and its output could be a second proximal

projection to me, on whose basis I could make the distinction after all. I take it that this adds nothing to the wide/narrow question, because I can sort on the basis of enhanced, composite sensorimotor projections in much the same way I do on the basis of simple ones.

The second way in which the distal distinction might be made would be in appealing to what I have *in mind* when I think of the safe mushroom and its tardively toxic lookalike, even if it were impossible to provide a test that could tell them apart. I don't have the same thing in mind when I think of these two kinds of things, so how can a narrow system that would assign them to the same category be a proper model for what I have in mind? Let us set aside this objection for the moment, noting only that it is based on a qualitative difference between what I have in mind in the two cases, one that it looks as if a narrow model could not capture. I will return to this after I discuss the third and most important sense in which we could make the distinction, namely, verbally, through language.

We could describe the difference in words, thereby ostensibly picking out a wide difference that could not be picked out by the narrow categorization model described so far.

7 Acquiring Asocial Categories Asocially, by Sensorimotor Toil

Let me return to the narrow model with the reminder that it was to be a *hybrid* symbolic/dynamic model. So far, we have considered only its dynamic properties: it can sort its proximal projections based on error-correcting feedback. But, apart from being capable of instrumental sensorimotor interactions such as eating, fleeing, mating, or manipulating on the basis of invariants it has learned by trial and error to detect in its proximal projections, such a system could also, as I had noted, simply assign a unique arbitrary name on the basis of those same proximal invariants.

What would the name refer to? Again, it would be an approximate, provisional subset of proximal projections that the system had learned to detect as being members of the same category on the basis of sensorimotor interaction, guided by error-correcting feedback. What was the source of the feedback? Let us quickly set aside (contra Wittgenstein 1953) the idea that acquiring such a "private lexicon" would require interactions with anything other than objects such as mushrooms and toad-

stools: in principle, no verbal, social community of such hybrid systems is needed for it to produce a lexicon of arbitrary category names and reliably use them to refer to the distal objects subtended by some proximal projections and not others (cf. Steels and Kaplan 1999; Steels 2001); there would merely be little point in doing so asocially. For it is not clear what benefit would be conferred by such a redundant repertoire of names (except possibly as a mnemonic rehearsal aid in learning), because presumably it is not the arbitrary response of naming but the instrumental response of eating, avoiding, and so on that would "matter" to such a system—or rather, such an asocial system would function adaptively when it ate the right thing, not when it assigned it the right arbitrary name.

But if the system should happen to be not the only one of its kind, then in principle a new adaptive road is opened for it and its kind, one that saves them all a lot of honest toil: the road of theft—linguistic theft.

8 The Advantages of Acquiring Categories by Symbolic Theft

At this point some of the inadvertent connotations of my theft/toil metaphor threaten to obscure the concept the metaphor was meant to highlight: acquiring categories by honest toil is doing it the hard way, by trial and error, which is time-consuming and sometimes perhaps too slow and risky. Getting them any other way is getting them by theft, because you do not expend the honest toil.

This is transparent in the case of Darwinian theft (which is perhaps better described as "inherited wealth"). In the case of symbolic theft, where someone else who has earned the category by sensorimotor toil simply *tells* you what's what, "theft" is also not such an apt metaphor, for this seems to be a victimless crime: whoever tells you has saved you a lot of work, but he himself has not really lost anything in so doing; so you really haven't stolen anything from him. "Gift," "barter," or "reciprocal altruism" might be better images, but we are getting lost in the irrelevant details of the trope here. The essential point is that categories can be acquired by "nontoil" through the receipt of verbal information (hearsay) *as long as the symbols in the verbal message are already grounded* (either directly, by sensorimotor toil or, indirectly and recursively, by previously grounded verbal messages)—and of course as long

as there is someone around who has the category already and is able and willing to share it with you.

9 Nonvanishing Intersections

Language, according to the model being described here, is a means of acquiring categories by theft instead of honest toil. I will illustrate with some examples I have used before (Harnad 1987, 1996b), by way of a reply to a philosophical objection to sensorimotor grounding models, the "vanishing intersections" objection: "Meaning cannot be grounded in shared sensorimotor invariants, because the intersection of the sensorimotor projections of many concrete categories and all abstract categories is empty." Perhaps the sensorimotor projections of all "triangles" and all "red things" have something in common—though one wonders about triangles at peculiar angles, say, perpendicular to the viewer, reducing them to a line, or red things under peculiar lighting or contrast conditions where the reflected light is not in the red spectral range—but surely the intersections of the sensorimotor projections of all "plane geometric shapes" vanish, as do those of "colored things," or "chairs," "tables," "birds," "bears," or "games"—not to mention the sensorimotor projections of all that is "good," or "true," or "beautiful".

By way of response I make two suggestions:

(1) Successful sorting capacity must be based on detectable invariance

The theorist who wishes to explain organisms' empirical success in sorting sensorimotor projections by means other than a detectable invariance shared by those projections (an invariance that of course need not be positive, monadic and conjunctive, but could also be negative, disjunctive, Plyadic, conditional, probabilistic, constructive, the result of any operation performed on the projection, including invariance under a projective transformation or under a change in relative luminance, indeed, any complex Boolean operation) has his work cut out for him—that is, if he wishes to avoid recourse to miracles, something a roboticist certainly cannot afford to do. Darwinian theft (innateness) is no help here, as Darwinian theft is as dependent on nonvanishing sensorimotor intersections as lifetime toil is. It seems a reasonable methodological assumption that if the projections can be successfully partitioned, then an

objective, nonvanishing, and detectable basis for that success must be contained within them.

(2) The invariance can be learned via experience or via hearsay

This is also the point at which linguistic theft comes into its own. Consider first the mushroom/toadstool problem: in a mushroom world I could earn these two important survival categories the hard way, through honest toil, sampling the sensorimotor projections and trying to sort them based on feedback from sometimes getting sick and sometimes getting nourished (Cangelosi and Harnad 2000). Assuming the problem is soluble, that is, that the projections are successfully sortable, then if I have the requisite learning capacity, and there is enough time in the day, and I don't kill myself or die of hunger first, I will sooner or later get it right, and the basis of my success will be some sort of invariance in the projections that some internal mechanism of mine has laboriously learned to detect. Let's simplify and say that the invariant is the Boolean rule "if it's white and has red spots, it's a toxic toadstool; otherwise it's an edible mushroom."

Life is short, and clearly, if you knew that rule, you could have saved me a lot of toil and risk if you simply *told* me that that was the invariant: A "toadstool" is a "mushroom" that is "white" with "red spots." Of course, in order to be able to benefit from such a symbolic windfall, I would have had to know what "mushroom" and "white" and "red" and "spots," were, but that's no problem: symbolic theft is recursive, though not infinitely regressive. Ultimately, the vocabulary of theft must be grounded directly in honest toil (and/or Darwinian theft); as mentioned earlier, it cannot be symbolic theft all the way down.

So far, we have not addressed the vanishing-intersections problem, for I preemptively chose a concrete sensory case in which the intersection was stipulated to be nonvanishing. Based on (1), above, we can also add that empirical success in sensorimotor categorization is already a posteriori evidence that a nonvanishing intersection must exist. So it is probably reasonable to assume that a repertoire of unproblematic concrete categories like "red" and "spotted" exists. The question is about the more problematic cases, like chairs, bears, games, and goodness. What could the sensorimotor projections of all the members of each of these categories possibly share, even in a Boolean sense?

10 The "Peekaboo Unicorn"

By way of a reply, I will pick an even more difficult case, one in which it is not only their intersection that fails to exist, but the sensorimotor projections themselves. Enter my "Peekaboo Unicorn": A Peekaboo Unicorn is a Unicorn, which is to say, it is a horse with a single horn, but it has the peculiar property that it vanishes without a trace whenever senses or measuring instruments are trained on it. So, not just in practice, but in principle, you can never see it; it has no sensorimotor projections. Is "Peekaboo Unicorn" therefore a meaningless term?

I want to suggest that not only is Pekaboo Unicorn perfectly meaningful, but it is meaningful in exactly the same sense that "a toadstool is a white mushroom with red spots" or "a Zebra is a horse with black and white stripes" are meaningful. Via those sentences you could learn what "toadstool" and "zebra" meant without having to find out the hard way—though you could certainly have done it the hard way in principle. With the Peekaboo Unicorn, you likewise learn what the term means without having to find out the hard way, except that you couldn't have found out the hard way; you had to have the stolen dope directly from God, so to speak.

Needless to say, most linguistic theft is non-oracular (though it certainly opens the possibility of getting meaningful, unverifiable categories by divine revelation alone); all it requires is that the terms in which it is expressed should themselves be grounded, either directly by honest toil (or Darwinian theft), or indirectly, by symbolic theft, whose own terms are grounded either by . . . etc.; but ultimately the buck must always stop with terms grounded directly in toil (or Darwin).

In particular, with the Peekaboo Unicorn, it is "horse," "horn," "sense," "measuring instrument," and "vanish" that must be grounded. Given that, you would be as well armed with the description of the Peekaboo Unicorn as you would be with the description of the toadstool or the zebra to correctly categorize the first sensorimotor projection of one that you ever encountered—except that in this case the sensorimotor projection will never be encountered because it is both unobservable and does not exist (a "quark" or "superstring" might be examples of unobservable-in-principle things that *do* exist). I hope it is transparent that the ontic issue about existence is completely irrelevant to the cogni-

tive issue of the meaningfulness of the term. (So much for wide meaning, one is tempted to say, on the strength of this observation alone.)

11 Meaning or Grounding?

At this point I should probably confess, however, that I don't believe that I have really provided a theory of *meaning* here at all, but merely a theory of symbol grounding, a theory of what a robot needs in order to categorize its sensorimotor projections, either as a result of direct trial and error learning or as a result of having received a string of grounded symbols: a symbol is grounded if the robot can pick out which category of sensorimotor projections it refers to. Grounding requires an internal mechanism that can learn by both sensorimotor toil and symbolic theft. I and others have taken some steps toward proposing hybrid symbolic/nonsymbolic models for toy bits of such a capacity (Harnad et al. 1991; Tijsseling and Harnad 1997; Cangelosi, Greco, and Harnad 2000), but the real challenge is of course to scale up the sensorimotor and symbolic capacity to Turing-Test scale (Harnad 2000b).

Imagine a robot that can do these things indistinguishably from the way we do. What might such a robot be missing? This is the difference between grounding and meaning, and it brings us back to a point I deferred earlier in this chapter, the point about what I have *in mind* when I mean something.

12 Mind and Meaning

According to the usual way the problem of mind and the problem of meaning are treated in contemporary philosophy, there are not one but two things we can wonder about with respect to that Turing-scale Robot:

1. Is it conscious? Is there anything it *feels* like to be that robot? Is there someone home in there? Or does it merely *behave exactly as if* it were conscious, but it's all just our fantasy, with no one home in there?
2. Do its internal symbols or states mean anything? Or are they merely systematically *interpretable exactly as if* they meant something, but it's all just our fantasy, with no meaning in there?

For (2), an intermediate case has been pointed out: books, and even computerized encyclopedias, are like Turing Robots in that their symbols are

systematically interpretable as meaning this and that, and they can bear the weight of such a systematic interpretation (which is not a trivial cryptographic feat). Yet we don't want to say that books and computerized encyclopedias mean something in the sense that our thoughts mean something, because the meanings of symbols in books are clearly parasitic on our thoughts; they are mediated by an external interpretation in the head of a thinker. Hence, on pain of infinite regress, they are not a viable account of the meaning in the head of the thinker.

But what is the difference between a Turing Robot and a computerized encyclopedia? The difference is that the robot, unlike the encyclopedia, is grounded: the connection between its symbols and what they are interpretable as being about is not mediated by an external interpreter; it is direct and autonomous. One need merely step aside, if one is skeptical, and let the robot interact with the world indistinguishably from us, for a lifetime, if need be.

13 Meaning and Feeling

What room is there for further uncertainty? The only thing left, to my mind, is question 1 above, namely, *while* the robot interacts with the world, *while* its symbols connect with what they are grounded in, does it have anything *in mind?* There is, in other words, something it *feels* like to mean something; a thought means something only if (a) it is grounded, directly and autonomously, in what it is otherwise merely interpretable-by-others as meaning *and* (b) it *feels* like something for the thinker to think that thought (Harnad 2000, 2001).

If you are not sure, ask yourself these two questions: (i) would you still be skeptical about a grounded Turing-scale Robot's meanings if you were guaranteed that the robot was conscious (feeling)? And (ii) would you have any clue as to what the difference might amount to if there were two Turing Robots, both *guaranteed* to be unconscious zombies, but one of them had meaning *and* grounding, whereas the other had only grounding?

Exercise: Flesh out the meaning, if any, of that distinction! (Explain, by hearsay, what it is that the one has that the other lacks, if it's neither grounding nor feeling; if you cannot, then the distinction is just a nominal

one, i.e., you have simply chosen to call the same thing by two different names.)

Where does this leave our narrow theory of meaning? The onus is on the Turing Robot that is capable of toil and theft indistinguishable from our own. As with us, language gives it the power to steal categories far beyond the temporal and spatial scope of its sensorimotor interactions and data. That is what I would argue is the functional value of language in both of us. Moreover, it relies entirely on its proximal projections, and the mechanisms in between them for grounding. Its categories are all provisional and approximate; ontic considerations and distal connections do not figure in them, at least not for the roboticist. And whether it has mere grounding or full-blown meaning is a question to which only the robot can know the answer—but unfortunately *that* is the one primal sensorimotor category that we cannot pilfer with symbols (Harnad 2000a, 2001b).

Notes

1. There are no doubt intermediate cases, in which "prepared" category detectors must first be "activated" through some early sensorimotor exposure to and interaction with their biologically "expected" members (Held and Hein 1963) before we can help ourselves to the category they pick out. There are also ways of quantifying how interconfusable the initial sensorimotor projections are, and how much toil it takes to resolve the confusion. For the most part, however, the distinction between prepared categories (Darwinian theft) and unprepared ones learned by trial and error (honest toil) can be demonstrated empirically.

2. Nets are also adept at unsupervised learning, in which there is no external feedback indicating which sensorimotor projections belong in the same category; in such cases, successful sorting can arise only from the pattern of structural similarities among the sensorimotor projections themselves—the natural sensorimotor landscape, so to speak. Some of this may be based on proximal boundaries already created by Darwinian theft (inborn feature detectors that already sort projections in a certain way), but others may be based on natural gaps between the shapes of distal objects, as reflected in their proximal projections: sensorimotor variation is not continuous. We do not encounter a continuum of intermediate forms between the shape of a camel and the shape of a giraffe. This already reduces some of the blooming, buzzing confusion we must contend with, and unsupervised nets are particularly well suited to capitalizing on it, by heightening contrasts and widening gaps in the landscape through techniques such as competitive learning and lateral inhibition. (Perhaps the basis for this should be dubbed

"cosmological theft" [Harnad 1976].) There are also structural similarity gradients to which unsupervised nets are responsive, but, being continuous, such gradients can be made categorical only by enhancing slight disparities within them. This too amounts to either Darwinian or cosmological theft. In contrast, honest toil is supervised by the error-correcting feedback arising from the consequences of the system's having miscategorized something, rather than from the a priori structure of the proximal sensorimotor projection.

7

Authentic Intentionality

John Haugeland

Editor's Note

What is the relation between computation and intentionality? Cognition presupposes intentionality (or semantics). This much is certain. So, if, according to computationalism, cognition is computation, then computation, too, presupposes intentionality (or at least some computations do). It follows that in order to understand what is required for computation (and cognition) we need to understand what is required for intentionality, and this is what Haugeland sets out to do. Starting with his previous division of intentionality into "original" and "derivative" (where "derivative" just means "not original"), he introduces the new tripartite distinction of "authentic," "ordinary," and "ersatz," the last of which, as the name suggests, is not genuine, but rather "intentionality-like." The other two (together with Haugeland's original distinction) form the four categories of "genuine" intentionality, which, by stipulation, presupposes the capacity for objective knowledge. With this stipulation, Haugeland intends to mark a crucial distinction between two kinds of intentionality to show what capacities are involved in and hence required for genuine intentionality. Focusing on scientific knowledge as an exemplary case of objective knowledge and the various processes involved in obtaining it, Haugeland uncovers what is required for objective knowledge, that is, for the possibility of nonaccidental objective truth: it is "the taking of responsibility on the part of the person (or 'system') who is capable of such objective knowledge." In particular, three kinds of self-criticism in scientific research, of increasing severity and consequence for a scientific field, figure crucially in the establishment of objective knowledge: "first-order self-criticism," the way scientists scrutinize experimental procedures to ensure that they are in accord with norms of proper performance; "second-order self-criticism," the scrutiny of the very norms that establish the various procedures; and "third-order self-criticism," the acceptance that basic laws and principles governing the domain of objects studied fail despite all attempts and efforts to make them work. These three ways of being self-critical about their own achievements and their field require scientists to accept increasing degrees of responsibility, culminating in what Haugeland calls "authentic responsibility" (in the case of third-order self-criticism). This kind of responsibility is authentic, because it is "personal responsibility for the science as a whole, including in particular the basic laws and principles in terms of which the objects are understood," and it is the ability

to accept this responsibility that underwrites scientific objectivity and hence is required for genuine intentionality. The implications for cognitive science (and, consequently, computationalism) are far-reaching: any system capable of authentic intentionality must also be capable of accepting authentic responsibility. Hence Haugeland sets a research agenda to figure out what it takes to implement the various kinds of responsibility if systems are to have genuine intentionality.

1 Computation and Intentionality

Once upon a time, we "knew" what computation is: it's whatever it is that Turing machines do—generalized, of course, to include von Neumann machines, production systems, LISP machines, and all the other provably "equivalent" architectures. By these lights, so-called analog computers weren't really *computers* in the strict sense, but only devices that had that name for historical reasons. Now, however, things don't seem so clear—hence the reason for this book. I will mention in passing just two of the numerous issues that have clouded the idea of computation in the last decade or two.

The first is the emergence and mathematical analysis of systems, with totally different architectures and very powerful capabilities, of which it is hard to say that they are not *computing* (in some pretheoretical sense). The most conspicuous among these are connectionist networks, artificial life (A-life) systems, and some dynamical models (including some that make essential use of chaotic attractors). It's enough to make one rethink the dismissal of old-fashioned analog systems. So, the question arises: what might it be that makes all of *these* (but not everything) *computing* devices?

And the second is the idea that maybe *semantics* is essential to computation as such. This is encouraged by the easy thought that to count as computing at all, a device must be computing *something* (and not just some "function," because, in some sense, everything does that). But there are also two more important motives for bringing semantics into the picture. One is the hope that semantics might be what is needed to answer the question raised by the first issue I mentioned: what makes all of *those* systems *computers?* And the other is the long-sought assimilation of computation to cognition—for cognition *surely* presupposes semantics.

I will not try to address either of these issues directly, but rather engage in what might be thought of as necessary spade work preparatory to a

resolution of either of them. In particular, I want to formulate a thesis about what is required for semantics—or, as I prefer to call it, *intentionality*—especially insofar as it is prerequisite to cognition. As I formulate it, this requirement will be neutral as to the various possible "architectures" for computation, but I don't know whether it really is neutral. That's probably an empirical question to which no one can yet know the answer.

2 Species of Intentionality

Almost twenty years ago, John Searle and I each introduced a pair of contrasting terms for two different species or types of intentionality. His distinction and mine are often confused, but they are not equivalent.

I distinguished what I called *derivative* intentionality from *original* intentionality. Derivative intentionality (or meaning) is the intentionality that something has only by virtue of having it *conferred* on it by something else that already has it. A common example is the intentionality of words and sentences, which (according to this view) they have only because it is conferred on them by thoughts, which already have it.

Original intentionality, by contrast, is any intentionality that is *not* derivative. I offered no positive account of original intentionality—what it requires or what makes it possible—but only the observation that there must *be* some, for the simple reason that not all intentionality could be derivative. My purpose was only to make it clear that original intentionality is the real problem, and to be able to raise the question of what has it and what it takes to have it.

Searle, on the other hand, distinguished what he called *observer-relative* intentionality from *intrinsic* intentionality. Observer-relative intentionality, at least insofar as it also means user-relative, turns out to be much the same as derivative intentionality, and, more recently, Searle has often adopted the latter term. Searle's original term, however, does have the following advantage: it suggests, correctly, that derivative intentionality is intentionality only to or for something else—a user or observer, for instance—never to or for whatever has it itself.

But intrinsic intentionality means much more than just original intentionality. Intrinsic intentionality is understood by Searle as an intrinsic, higher-order property that some physical structures have just in virtue of

their (intrinsic) physical structure. So it's supposed to be analogous to the wetness of H_2O (water) or the hardness of tetrahedrally crystalline carbon (diamond). I, like many other philosophers and cognitive scientists, have had a hard time taking the notion of intrinsic intentionality seriously. But it does have one important merit that the notion of original intentionality lacks: it is intended as a *positive* account of the source ("origin") of all other intentionality. The fact that Searle's discussions of it are sorely lacking in plausible detail about what makes it possible does not cancel out *this* merit.

What I now want to do is offer *my own* positive account of what original intentionality requires and what makes it possible. My account will be profoundly different from Searle's and will include significantly more detail. But one way in which it will not differ from his is that it will not contain any hints at all about exactly what it takes to *implement* it. Thus, though more detailed in various ways, it is like Searle's in being, in effect, a challenge to empirical science: go figure out what it would take to implement *this*.

To explain my proposal, I will need to introduce a new classification of types or species of intentionality (or "intentionality-like" phenomena). This is a three-way classification, for which I will use the following terms:

1. *Authentic* intentionality
2. *Ordinary* intentionality
3. *Ersatz* intentionality

None of these is equivalent to any of original, derivative, intrinsic, or observer-relative intentionality. The relations among the old and new types go like this: either authentic or ordinary intentionality can be either original or derivative (yielding four possible subtypes, if you like). I will refer to all of these generically as *genuine* intentionality. Ersatz intentionality is none of the foregoing, but rather, as the name suggests, not genuine intentionality at all; it is only an imperfect analogue that is in some ways "intentionality-like"; and, finally, there is no such thing as intrinsic intentionality.

The relation between authentic and ordinary intentionality is more complicated. In general, only systems that are in some sense *capable of* authentic intentionality are *capable of* ordinary intentionality. (I say "in general" because there may be deficient or partial cases that would call for tedious qualifications.) But *actually having* ordinary intentionality

does not require *actually having* authentic intentionality. One of my main aims here is to make the reasons for this complicated claim clear.

Before proceeding to that, however, let me explain briefly what I mean by ersatz intentionality (and then be done with it). Ersatz intentionality is the so-called intentionality that can be attributed in what Dennett calls "the intentional stance." I don't mean that *all* intentionality attributable in the intentional stance is merely ersatz, but rather that the most that such attributability *suffices for* is ersatz intentionality. In other words, in my view, the intentional stance by itself is not useful at all for explicating the *difference* between genuine and ersatz intentionality.

In particular, I maintain that the intentionality-like character of various states attributable to (subhuman) animals, or (so far as I know) to any actual or concretely envisioned robots, is not genuine intentionality but merely ersatz. Examples like thermostats and flowering trees are just ludicrously extreme cases. I will not explicitly defend either of these claims (about the intentional stance or about animals and robots), but my reasons for them should become obvious as the story unfolds.

3 Intentionality and Objectivity

The first point I want to make about genuine original intentionality, whether ordinary or authentic, is that it presupposes the capacity for objective knowledge. Indeed, since derivative intentionality presupposes original intentionality, the argument will apply to *all* genuine intentionality. In the present context, I mean by 'objective knowledge' beliefs or assertions that are true of objects nonaccidentally. That the intentionality of beliefs and assertions presupposes the possibility of some of them being objectively true—that is, true of objects as they actually are—is obvious. For, if none of them could even possibly be true of actual objects, it would make no sense to say that any of them were about or intended any objects at all (or, therefore, to call any of them beliefs or assertions at all).

It is equally obvious that beliefs or assertions that are genuine knowledge must be not only true but true *nonaccidentally*. What this means, however, and how it is achieved scientifically, is not so obvious and will emerge in what follows in several stages. But that the capacity for such knowledge is prerequisite for *genuine* intentionality is true by stipulation: it's what I mean by "genuine." The point of the term, after all, is to mark

a distinction—a distinction that is philosophically important and that goes missing in those discussions that indiscriminately run together the "intentionality" of people, animals, and (current) robots (not to mention flowering trees and thermostats). The aim of the arguments to follow is not to defend this stipulation but rather to show how much it involves—that is, how deep a gulf the distinction it marks really is.

(In the case of other propositional attitudes and speech acts—such as desires, doubts, commands, questions, and so on—the connection between intentionality and the possibility of objective truth is less direct but just as essential. I expect this claim to be uncontroversial; but, in any event, I will take it for granted without argument. I believe that an analogous case could also be made for nonpropositional intentional states—if there are any—such as mental images or internal emulations or simulations; but this, too, I will leave aside here.)

So, the fundamental question is this: what is required for the possibility of nonaccidental objective truth—that is, knowledge? I am going to argue that it requires the taking of *responsibility* on the part of the person (or "system") who is capable of such objective knowledge. Indeed, it will turn out that two levels of responsibility are involved; and it will be the difference between these levels that accounts for the distinction between ordinary and authentic intentionality. But first things first.

4 Objectivity and Responsibility

In order to make out the case for this essential relevance of responsibility, I will focus on a distinctive species of objective knowledge, namely, scientific knowledge. The reason is not just that scientific knowledge is especially explicit and compelling, but rather that it has been much studied, so its features and factors are fairly well known. But I am convinced that (perhaps with more work and less clarity) basically the same case could be made for common-sense knowledge. Thus, though I don't agree with Quine that science is "continuous with" common sense, I do agree that they have the same fundamental character. Accordingly, I maintain that examining the conditions of the possibility of scientific intentionality is a suitable entry point for examining those of any genuine intentionality. The advantage of the focus on science is purely expository.

One conspicuous fact about reputable scientists is that they are always highly trained and highly disciplined. Hand in hand with this conspicuous fact goes another: scientific research is always and inherently self-critical—not only individually but interactively. Such self-critical discipline is quite visible, for instance, both in how scientists perform experiments themselves and in how they evaluate each other's performances.

So we see here, what is obvious anyway, that scientific research is fundamentally communal, and, in particular, that it is governed by *communal norms of proper performance*—that is, of proper procedure and technique—in terms of which particular performances can be critically judged. These norms effectively define the *discipline* of normal scientific research. At the same time, they provide the basis for what I will call *first-order* scientific self-criticism. Disciplined science is self-critical in that it carefully scrutinizes actual procedures to ensure that they are in accord with its norms of proper performance.

This critical self-scrutiny is manifestly a crucial factor in the ability of scientific research to discover the truth about the objects it investigates, since it weeds out experimental results that are compromised by sloppy or improper procedures. Such results are, of course, unreliable—in the specific sense that they are either false or, if true, only so by accident ("dumb luck"). Therefore, first-order self-criticism is also essential to the fact that scientific results can be not only true but nonaccidentally so—that is, objective knowledge. Hence it is prerequisite to genuine intentionality.

There is also, however, a second sort of self-criticism in normal experimental practice. This comes into view when we remember that, unless a result is utterly routine, it usually won't be accepted until it has been replicated by other teams in other laboratories. But what does it mean to say that a result has been replicated, and why does it matter? Why, exactly, is there anything *wrong* if replication efforts fail? These questions may seem trivial, but I think they cut pretty deep.

Note first that replication of an experimental finding is seldom simply a duplication of the previous experiment. On the contrary, it's much better if the same result can be obtained by other means. So, our first question comes to this: how can a *different* experiment amount to a replication of (or a failure to replicate) the *same* result?

The answer is that (except, perhaps, in times of crisis) scientists always have pretty determinate convictions about what the objects they're investigating *must* be like. The most basic of these convictions are expressed in scientific laws and principles. These laws (and their more specific implications), in determining the objects, sharply constrain how the results of various different experimental procedures would *have to* be related.

To take a contrived example, suppose an experiment were performed to measure the electric charge on some newly isolated particle. This experiment will rely on certain laws about how charged particles interact with various other particles, fields, and so on. But there are lots of such interactions and combinations thereof. So another team could proceed in quite a different way and still be confident that it was measuring the same property of the same particle—that is, the same object, as determined by the same laws. In this way, the constraints expressed in the laws make it possible for two quite different experiments to produce the same result—or, as the case may be, a conflicting result.

In the light of all of that, however, it's also clear why replication—or, more to the point, a failure of a replication—*matters*. Inasmuch as the norms of proper performance, together with the laws governing the objects under investigation, effectively promise that the results will agree, any disagreement means there's something wrong somewhere. Of course, the most common upshot is that, on closer examination, one or the other of the experiments was not properly performed after all. Most apparently conflicting results are actually due to experimental error.

But the more interesting case is when no such error can be found. Then scientists turn their scrutiny not on individual performances but on the very norms that determine procedural propriety for all performances of that sort. This is not as drastic as it sounds: the business of refining observational technique in the light of experience is bread and butter to any thriving experimental practice.

In other words, empirical science is inherently self-critical in a second and deeper way: it critically scrutinizes not only individual performances but also the very practice itself. I call this second and deeper scrutiny *second-order* self-criticism. And, as with first-order self-criticism, this too is obviously a basic factor in the ability of scientific research to discover—nonaccidentally—the truth about its objects. Hence it too is prerequisite to the possibility of scientific knowledge and hence genuine intentionality.

So far, I have spoken about how scientific research as such is inherently self-critical, and I have distinguished two "orders" of that self-criticism. Further, I have pointed out that both the impetus and the need for this self-criticism arises out of the relation between scientific results and their objects—namely, the objects as what are determined and constrained by the basic laws and principles. In particular, it arises from the fact that certain combinations of results can show that something is wrong somewhere and so needs to be found and corrected. It is by giving content to the idea of being wrong that this basic structure contributes essentially to the possibility of being correct. Finally, I have drawn the obvious conclusion that whatever is prerequisite to getting the relation to objects correct is prerequisite to objective knowledge, and hence to genuine intentionality.

What I have not yet done is say anything about what scientists themselves must be like. Of course, it is the scientists who must "respond" when something is wrong, and a corrective response is required—it is they who must be "self-critical." But what must they be like, *qua scientists,* if this "requirement" is to get a grip on them, and move them to action?

Clearly, it must be something like this: scientists qua scientists will not and cannot tolerate a body of results in which there is something wrong in the way that I have been discussing. For such tolerance would undermine the process of weeding out unreliable results, and thereby also the ability to settle nonaccidentally on the truth about objects—that is, to discover it. Incompatible results, as determined by the laws and principles governing the objects under investigation, must be *unacceptable* to scientists qua scientists. Thus, in order to be a scientist, a person must be such that she will not—because she *cannot*—accept or tolerate results that are incompatible, hence somehow wrong.

Two clarifications are needed to ensure that this point is not misunderstood. First, the refusal to accept the results in question can't simply be a matter of denying or suppressing them. Rather, it must take the form of an effort to find out what went wrong and then correct it. Second, the "cannot" (in "cannot accept or tolerate") must have a normative rather that an alethic force. That is, it's not that the scientist is *unable* to tolerate such results (as in an inability to tolerate pain or pomposity), but rather that such tolerance would be *impermissible* in a scientist—incompatible with being a *genuine* or *honest* scientist.

But as soon as the point about unacceptability is understood in this way, it becomes clear that it can also be expressed as follows: being a scientist entails a certain kind of responsibility. In particular, it is the responsibility of any scientist, qua scientist, not to accept results in which something has been shown to be wrong, and, moreover, to regard finding and correcting whatever is wrong as incumbent on the scientific community.

5 Responsibility and Cognitive Science

If you gather up (and swallow) everything I've said so far, then this conclusion about science and responsibility has important implications for *cognitive* science. For I have argued that:

- genuine intentionality presupposes the capacity for objective knowledge;
- scientific knowledge is an exemplary case of objective knowledge;
- scientific research, as a route to objective knowledge, is inherently self-critical in at least two ways; and
- in order for research to be thus self-critical, scientists themselves must accept a certain kind of responsibility.

But those together imply that any system capable of genuine (original) intentionality must be capable of accepting that kind of responsibility. Hence any system capable of genuine *cognition* must have that same capacity for responsibility.

It is my impression that cognitive science—and especially cognitive science inspired by the idea of computation—has been effectively oblivious to this essential connection between cognition and responsibility. I suspect that an inkling of this oblivion is what lies behind various ill-formed (and often annoying) "intuitions" to the effect that AI systems (of whatever stripe) can't really *mean* anything, can't *understand* their own inputs and outputs, can't really be *conscious,* and so on. It's not that I think these intuitive judgments are false, but rather that they don't get at the underlying problem and hence remain *mere* intuitions, with regard to which "arguments" (pro and con) can be, at best, aggravatingly inconclusive.

Correlatively, I believe that this same oblivion also enables a profoundly misguided and misleading assimilation of human cognition to so-

called animal cognition. This assimilation then encourages the perverse supposition that, insofar as robot capabilities approach those of animals, they thereby also approach those of people. So far as I know, no (subhuman) animal, nor any current or envisioned robot, is capable of accepting any responsibility on its own. Therefore, by the above argument, none is capable of any genuine intentionality or cognition. This is why I have coined the term ersatz intentionality for those aspects of animal and/or robot behavior that are, in certain superficial ways, undeniably intentionality-*like*.

I believe that the foregoing argument suffices to show that at least most of contemporary cognitive science seriously underestimates its own essential field, and that it cannot fully succeed in its own goals unless it rectifies that underestimation. For, in being oblivious of personal responsibility as prerequisite to the possibility of original intentionality, which is acknowledged as prerequisite to genuine cognition, it denies itself the possibility of adequately understanding the latter.

Moreover, insofar as it is held that at least some species of computation presuppose genuine intentionality (a view on which I will voice no opinion), current approaches to understanding computation suffer from the same debilitating underestimation. (Such a view of computation would be required of anyone who held that cognition *just is* computation of a certain sort.)

All the same, I also believe that the discussion of intentionality so far remains essentially incomplete. In beginning to address this incompleteness, I will, to be sure, be raising the bar still higher for cognitive science (and perhaps for computer science). But I would like to reemphasize something that I indicated at the outset: it is no part of my aim to argue that the requirements that I am setting cannot be met. They are not intended as show-stoppers, still less as refutations, but rather as serious scientific challenges to fundamental research. Of course, I'm also in no position to claim that these challenges *can* be met—nor do I have any clear sense of what it would take to meet them. The best I can do as a philosopher is to try to articulate some of the boundary conditions.

In characterizing the second-order self-criticism that belongs to science, I observed that the norms of scientific practice *together with* the laws governing the objects under investigation "promise" that properly obtained scientific results will all agree about those objects. It is the breach

of this promise, when results conflict, that shows that *something* is wrong. But then I considered only one of the two obvious options for *what* might be wrong—namely, something in the norms governing the practice (that is, proper procedures). Scrutinizing and refining these is what I called second-order self-criticism.

Clearly, however, there is no guarantee that any way of refining and improving the practice (the norms) will actually eliminate all conflicts among properly obtained results. In other words, there may be no way to make this field, as presently constituted, "work." Thus, eventually, if the conflicts prove serious and persistent enough, then—and only then—scientists will begin to take their last option seriously: that there is something wrong with the basic laws and principles in terms of which they understand the objects in their domain.

This process I will call—predictably—*third-order* scientific self-criticism. It is always the *last* option because it raises the stakes in a fundamental way.

6 Authentic Responsibility

Revising the laws that a scientific community takes to govern the objects in its domain is a far more radical undertaking than merely revising the practices for investigating them. The reasons for this can be summarized in three closely connected points.

First: the laws and principles tend to be interdependent, forming a coherent whole or "package" such that you can't revise just one or two at a time, leaving the others as they were. Thus, any such revision is likely to have to be more or less "wholesale." By contrast, revisions in the norms for proper experimental procedure can usually be managed roughly at "retail"—that is, one or a few at a time.

Second: the experimental procedures, as we have seen, are themselves intelligible only in terms of the laws governing the objects. So, although you can refine procedures without revising laws, you can't revise laws without revising procedures—or at least reconceiving them. What's worse, since the law revisions are apt to be wholesale, the required procedural revisions will have to be more widespread as well. In other words, a whole lot of things have to be reconceived at once, which makes the space of alternatives much larger and the number of "fixed points" to guide the search much smaller.

But finally, and most fundamentally: the laws governing the objects are crucial to the very intelligibility of those objects as what they are. The point is most easily seen by example: how could one possibly understand what mass, force, momentum, and energy—not to mention electrons, orbiting planets, springs, and gasses—*are,* apart from how they figure (in relation to one another) in the relevant laws of motion, gravity, elasticity, thermodynamics, and so on? The connection between laws and intelligibility is conspicuous as well in the fact that scientific explanation, which is the context in which scientific understanding is made explicit, standardly appeals to laws and principles (or something like them).

Of course, the intelligibility of the objects also depends on the (various) empirical procedures by which they can be observed and measured—a truism that once lent plausibility to verificationism, operationalism, and their ilk. But, without the laws, even the procedures would make no *systematic* sense, and all that would remain would be a traditional craft (and perhaps some mythology). It is because the laws and principles *collectively* determine the intelligibility of the objects that they can't be revised piecemeal, nor without also revising the practice.

In sum, when it turns out (after long struggle) that a scientific discipline cannot in fact be made to work—too many of the results that it itself deems proper are, by its own lights, impossible—then that discipline as a whole comes into question. This is why I said that third-order self-criticism raises the stakes in a fundamental way. Often enough, the eventual changes are so radical that it makes as much sense to say that the old discipline died out and got replaced by a successor (or successors) related only by a pattern of family resemblances to what preceded. Hence the appeal of such grand phrases as "paradigm shift" and "revolution."

Yet, however radical, and however described, if a discipline just doesn't work, pursuing such transformations (or replacements) can sometimes be the only responsible response to the actual situation. Accepting this responsibility is peculiarly personal, not merely because it is so risky, but because what is at stake in it is the individual's own professional self-understanding. Who, after all, are you, professionally, if your professional specialty dies?

However, even though the ability of *some* scientists to accept this responsibility is essential to science—lest it degenerate into a dogmatic and unempirical orthodoxy—it is not essential that *every* scientist actually be able to accept it. As Kuhn argued, on historical grounds, most scientists won't

actually jump ship until they can clearly see a plausibly better vessel on the horizon. Revolutions, as everyone knows, always need their heroes.

Since this responsibility is, in some sense, for the whole, and yet peculiarly personal, I call it (not ordinary but) *authentic* responsibility.

7 Authentic Intentionality

Intentionality is directedness to *objects*—objects as they *are*. It therefore presupposes not only the distinction between objective truth and falsity, but also the ability to discover the former nonaccidentally—that is, to *know* it. We have seen this structure already in the conditions of the possibility of ordinary intentionality (first- and second-order self-criticism, and ordinary responsibility).

But what are *objects?* What does it mean to say "objects as they *are*"? In speaking to these questions, I will rely on an approach to metaphysics that is far from universally endorsed. Accordingly, my conclusions will presuppose some controversial premises. But, since I endorse this approach, I think the conclusions are valid. Your mileage may vary.

This is not the place to spell out my approach to metaphysics in detail, let alone defend it. I will mention only one (familiar but tendentious) claim that motivates it, and then the basic shape of an accommodation of this claim. The claim is that "absolute" or "capital-R" Reality—Reality as *God* describes it—is a notion that cannot be made sense of. Note: this is not the claim that we can never come to know "capital-R Reality," but rather the claim that the notion of such reality does not make sense, and that therefore neither does knowing it or not knowing it. The philosophical challenge, then, is to make sense of objectivity, truth, and knowledge without appealing to any notion of "capital-R Reality."

The basic shape of the response is to relativize reality not to God's descriptive resources but to our own—the only ones we can ever have or understand. Then, of course, the demon to be exorcised is mere or capital-R *Relativism,* which is incompatible with any robust conception of objectivity. What's so upsetting about such relativism is that, according to it, our descriptive resources are ultimately arbitrary—in effect, accidental—the results of personal taste, historical happenstance, extraneous cultural influences, and the like. This is really just the flip side of capital-R Realism, since it accepts the same basic assumption that the only alter-

native to arbitrariness is capital-R Reality. Therefore, what any more satisfactory relativizing must do is show how the descriptive resources in question are not ultimately arbitrary.

But that, after all—showing how descriptive resources are not ultimately arbitrary—is exactly what *authentic* responsibility is all about. It is authentic in that it is personal—one's *own*—responsibility for the science as a whole, including in particular the basic laws and principles in terms of which the objects—hence the terms for them—are understood. In other words, it is responsibility for the basic descriptive resources of the field. But, inasmuch as it is genuine *responsibility,* it is not arbitrary.

Third-order self-criticism is driven—neither easily nor happily—by two factors. It is driven by an overarching insistence on maintaining a discipline that actually works by its own demanding lights—that is, a discipline that reliably produces stringently constrained results that are consistently possible according to its own defining laws and principles. And, it is driven by a persistent empirical recalcitrance in those results—results that have themselves all survived careful first- and second-order critical scrutiny.

In other words, it is driven by a persistent empirical failure to make the discipline work, despite assiduous and often ingenious efforts on the part of many. Whether a discipline can be made to work is not up to the scientists (or history or fads or culture). There's nothing *arbitrary* or *accidental* about it. That's why I can say that, in the end, giving the whole thing up is not just a matter of personal preference or social trends, but rather of the highest scientific *responsibility*.

Accordingly, the ability to accept this responsibility—*authentic* responsibility—on at least some occasions by at least some scientists, is prerequisite to scientific objectivity. Hence, by my earlier argument, it is prerequisite to genuine intentionality and cognition. I call the intentionality of someone who does accept authentic responsibility *authentic intentionality*.

And that is why I said earlier (in section 2) that the *capacity* for authentic intentionality is prerequisite to the *capacity* for ordinary intentionality, but *actually having* the former is not prerequisite to *actually having* the latter. But even this somewhat complicated dependency shows that cognitive science will not have understood the conditions of the possibility of intentionality—hence of cognition at all—until it has understood the possibility of authentic responsibility and intentionality.

8 Dramatic Conclusion

It used to be said—perhaps reassuringly, perhaps defensively—that the aims of artificial intelligence are limited in a certain way. The goal, it was said, is not to construct a machine (or "system") capable of the full gamut of human experience or of the human mind, but rather only a system capable of human-like *intelligence* and hence cognition (so far as it is required for intelligence).

The intended contrast (what AI is not concerned with) included things like emotions (love, anger, fear, . . .), feelings (pleasure, disgust, embarrassment, . . .), and perhaps nonintellectual evaluations (this is fun, fascinating, beautiful, . . .). I'm inclined to think that none of these separations—topical quarantines, in effect—will ultimately prove defensible. But here I want to broach just one such doubt.

I have argued that the objectivity and intentionality of scientific thought—and, by extension, all thought—depends essentially on a certain rather rich structure of self-criticism and responsibility. The deepest and most fundamental layer of this structure—authentic responsibility—can be characterized generically as follows: it is an *honest commitment*—in the sense of resolve or dedication—to making something work, on pain of having to give the whole thing up. Such honest commitment is "double-edged"—it cuts both ways—in that, first, it requires honest and dedicated effort to making it work, and yet, second, it also requires the honest courage, eventually, to admit that it can't be made to work—if it can't—and then to quit. These two edges are ordinary and authentic responsibility, respectively.

But I think that this structure of double-edged commitment is widespread in human life—not just in science, but in many (and in some sense all) areas of the greatest human import. In particular, I believe that it is the basic structure of love, and also freedom, and that love and freedom are its most basic forms. Therefore, I close with the following dramatic—but also perfectly serious—claim: cognitive science and artificial intelligence cannot succeed in their own essential aims unless and until they can understand and/or implement genuine freedom and the capacity to love.

Epilogue

The notion of computation has shaped our way of thinking about the mind. From the daring imaginations of the early mechanists to the present computational models of artificial intelligence and cognitive science research, the idea that mind can in some sense be understood in mechanistic terms—as some sort of machine—has permeated the investigations and explanatory attempts of the mental.

Where are we left at the beginning of this new century in the foundations of cognition? It seems to me that what mind is and exactly how it is brought about by physical systems is to a large extent still as much a mystery as it was fifty years ago, when computationalism provided a new foundational tenet for scientific investigations of the mind. I do not intend to be negative about the achievements of computationalist cognitive science; quite the contrary. Nor am I pessimistic about the future of cognitive science research. Neither do I want to advocate a "mysterian" view, which takes minds as intrinsically inaccessible to scientific inquiry, as nonobjective, mere subjective phenomena that cannot be tackled by scientific methods in principle. Rather, I am somewhat skeptical about the effectiveness and adequacy of the conceptual apparatus currently employed in our investigations of cognition. That is, I suspect that we are still lacking the appropriate conceptual tools to grasp the intricate relationships between mind, brain, and body—a conceptual void not restricted to the domain of the mental, but very visible in the domain of computing as well.

Computation as Real-World Activity

The struggle of computer science with its own conceptual foundations is apparent in many ways, but nowhere as clearly as in its educational incarnation, that is, the recent discussions about necessary revisions to current computer science curricula. Whereas up to the end of the last century these curricula were still largely dominated by the view that computation is Turing-machine computation—what else could it be?—we are now witnessing a shift in the educational directive. The deeply seated conviction that computation is "the computation of a function" has given way to new approaches that view computations as interactive processes. And what used to be core courses of a computer science curriculum (such as "automata," "formal grammars," "computability theory," etc.) might very well become electives. Fundamental revisions of that sort to a formal educational curriculum are indications of a discipline's having acknowledged (in a Kuhnian sense) *qua* discipline its own previous foundational confusion, that it—to a large extent—misled itself regarding what it is all about.

With computer science maturing and computation becoming part and parcel of our daily lives, we are beginning to grasp the impact of computational systems and computational metaphors on our way of thinking. All kinds of systems, including cognitive systems, have become the target of various ways of computational conceptualizing; not always rightfully so, however.

Some of the reservations about applying computational concepts to cognitive systems voiced by philosophers, connectionists, dynamicists, and others are certainly justified. Today it seems clear, for example, that classical notions of computation alone cannot serve as foundations for a viable theory of the mind, especially in light of the real-world, real-time, embedded, embodied, situated, and interactive nature of minds, although they may well be adequate for a limited subset of mental processes (e.g., processes that participate in solving mathematical problems). Reservations about the classical conception of computation, however, do not automatically transfer and apply to real-world computing systems. This fact is often ignored by opponents of computationalism, who construe the underlying notion of computation as that of "Turing-machine computation." Computers are not abstract, but physical systems (consisting of

monitors, keyboards, disk drives, etc.) that need to be controlled by programs. Hence issues of timing, uncertainty, reliability, and effects typical of dynamic systems like "coupling," "phase locking," or "continuous reciprocal causation," which dynamicists take to be essential to mind, are just as much essential to computation, at least at some levels. At the level of VLSI chips design, for example, engineers have to deal with a multitude of physical processes that happen simultaneously in the chip. These processes have to be carefully controlled in order for them to support a particular abstraction that is "reliable," "clocked," "digital," "discrete," and "serial"—the computational processes as we know them from the perspective of a higher programming language. Just because at that level we do not have to worry about low-level timing issues related to latency times in reading and writing registers on disk controllers, keyboard-sampling routines, sound-producing processes, and so on does not mean that these issues are not there or can be ignored.

On a larger scale, any controller of an embedded system has to cope with such issues. In artificial intelligence research, "situated AI" has been the major driving force supporting the view that we need to think of intelligent systems as *embodied* (i.e., having a body to control, which is part of the world) and *embedded* in their environment. The offspring is a research strategy that recognizes the fruitful repercussions on a theory of cognition resulting from building complete systems: at the very least the intrinsic constraints and limitations imposed on intelligent systems by real-world dynamics will eliminate theoretically possible but practically infeasible solutions. As an aside, the importance of practical feasibility has long been acknowledged in theoretical computer science in the shift from recursion theory (what can be computed in principle) to complexity theory (what can be computed in practice).

Computationalism—New Directions

Once the restricting, classical view of computation as abstract, discrete, syntactic, disconnected, and halting is abandoned in favor of a more encompassing perspective that allows computations to be concrete, continuous, semantic, connected, and ongoing, we will be in a position to redefine the computationalist paradigm from a new perspective. This book is a

first attempt to sort out at different levels some of the territory that will have to be covered by such a successor notion.

Theoretically, intentionality will not be the only contestant in the arena of concepts on which computation may depend; at the very least, it will be joined by the notion of responsibility (chapter 7). And once the possibility is entertained that "theory of computation" is a misnomer, because computation per se does not constitute an intellectually interesting, coherent subject matter (chapter 2), then we may be more willing to concede that there is no single such thing as computation (and hence no right definition of it). Furthermore, by taking issues of embodiment and situatedness seriously, we may be able to construct a narrow notion of meaning that is grounded in the interaction of a system with its environment and can serve as the venture point for a theory of human language (chapter 6).

Practically, Turing computability seems to be a notion of little relevance for the realm of computing. Hence cognitive science (and philosophy) should not cling to it. For one, historically it did not play any role in the conception and realization of computing systems as we know them (chapter 4), and furthermore, it might not be a theoretically relevant limit to cognition that delineates a boundary that minds cannot transcend (chapter 3). Once researchers of the mind recognize and acknowledge the tensions imposed by the various theoretical schemes (chapter 5), alternative conceptions of computation (from quantum to DNA computations) may become accepted outside computer science, in particular, cognitive science, as part of the standard computational repertoire.

Computation as Experimental Philosophy

Examples from computational practice may shed new light on many of today's unresolved issues in cognitive science and the philosophy of mind (such as questions about "causation," "consciousness," "mental content," "qualia," and others). Although often not obvious prima facie, computing systems deal with many problems similar to those that minds have to cope with on a regular basis. Yet, as opposed to minds, computers are systems we build. Hence computational examples may elucidate otherwise opaque and obscure relationships.

For example, simple instances of the "other minds problem" come up regularly in communication situations between two computers, where

one computer cannot be sure whether the message it sent to the other computer has been received or "understood." Nor could the computer ever "know" whether the message was indeed received by a machine of its kind. In fact, often computers are "deceived" because we purposefully design software to make one system "think" it is communicating with a system of its kind (e.g., think of "terminal emulations").

In a similar vein, simple instances of Cartesian skepticism are implemented in the startup sequence of computer systems, where the operating system has to figure out what kind of machine it is running on, how much physical memory and what kind of permanent storage are available, and so on. Software engineers devise clever algorithms to detect the available hardware, but in the end the program is restricted to its own "epistemological resources" (as apparent in the case of an operating system running on a virtual machine, which emulates a system different from the one it is running on—the operating system has no way of "knowing" or "noticing" that fact).

Another example concerns the relationship (often called "supervenience") between different levels of description or layers of organization in nature. The idea is that while upper layers are determined and dependent on lower layers, upper layers are not necessarily reducible to the lower ones. Many software systems exhibit complex layered architectures with rich interactions and dependencies among various layers that may serve as examples of such dependency relations. They may also be useful as models of virtual machine architectures (e.g., as implemented by the brain) and may lead to a clarification of notions like "mereological supervenience" and "emergence," possibly demarcating the path toward a general theory of abstraction, which defines criteria for all "legitimate" abstractions given a level of description.

Many other psychological and philosophical problems can be studied using examples from computing: questions of causation (upper-level as well as upward/downward causation), the impact of space-time constraints and limited resources on processing mechanisms, effects of learning and adaptation on architectural organization, the nature of qualitative states and their architectural requirements, the origin of representation and reference in systems (e.g., how digitality is supported by dynamic systems that stabilize over spatiotemporal regions and how the resulting digits can be used as representations), different ways of repre-

senting external and internal states, architecture-based concepts, and many more.

Toward a Successor Notion of Computation

One can never be certain of what will turn out to be relevant for future developments in any field, but I am still tempted to make a prediction about computation, that is, what ingredients will be part of (the definition of) a *successor notion of computation*. First and foremost, I am convinced that three notions of already increasing importance in computational practice will figure crucially in future versions of computation: the notions of distributed computing, resource limitation, and locality. These three notions exhibit interesting three-way dependencies. Because of resource limitations (e.g., limited space), local computations often need to be distributed. Yet, there are limits to what can be distributed, again because of resource limitations (e.g., time lags between spatially distant computers), so the explicit acknowledgement of what is local and what is remote becomes important. By allowing the notions of time and space to enter explicitly into computational descriptions—not only in terms of discrete orders, but as continuous metrics that model actual distances in time and space—computational descriptions will be able to reflect real-world constraints in much the same way that differential equations reflect the underlying metrics of the dynamical systems they capture. For example, a program running on a node in a distributed system may be able to determine online whether it pays off timewise to involve remote nodes in an ongoing computation if it has access to the durations of various operations on those nodes as well as information about their spatial distances.

Especially with the advance of "mobile computation," the locality of a computation has become an issue—up to now computations have usually taken place in one system. Even if they use remote resources (e.g., through remote procedure calls), this simply means reaching out temporarily, without leaving the primary locus of computation. Hence there is no need to acknowledge the *hic et nunc* of the computational process. Mobile computing, however, requires new control mechanisms to deal with the locality, mobility, and distribution of computations, which in turn intro-

duce new limiting factors such as latency and bandwidth that computations need to deal with explicitly.

With the distribution of processing, related issues of parallelism, synchronization, and others will become even more important, and the respective concepts will have to be augmented to address spatiotemporal constraints. More than before, we will have to deal with "emergent effects," which inevitably show up in large, distributed computational systems, and study possible ways to control them (e.g., the flooding of subnets on the internet with useless information caused by computer viruses in an effort to block the entire traffic to particular servers in that subnet).

Explicitly addressing resource limitations, for example, may prevent some negative emergent systemic side effects (such as the emergence of "thrashing" in an operating system). In general, properties of distributed processing and ways of controlling emergent behavior in computing systems may give us new ideas about what kind of distributed control is implemented in brains and how brains manage to keep in control of their parts, for the most part (that brains are not always able to maintain global synchronization is apparent from various kinds of epileptic attacks and seizures, spontaneous loss of consciousness, and various other disturbances).

Another element figuring crucially in the conceptual framework of a successor notion of computation is the notion of interaction, for computing systems are not viewed in isolation, but as part of an integrated web of computational activity. The crucial question, then, is how to understand the notion of interaction and how to describe interactive systems. For example, it seems clear that code written for interactive systems will not be comprehensible by itself: the purpose of the program

`while true do` write(read(in_channel),out_channel) **`endwhile`**

is unclear, and hence *what kind of program it is,* if viewed in isolation of the internet packet router it is running on. Other questions will concern the very nature of interaction and its different realizations (e.g., whether bandwidth limitations and problems of noise will lead to reliance on knowledge-based communication typical of human communication, where the communication works only because of large bodies of preexisting shared knowledge).

The most intriguing and consequential pair of notions for future investigations of the mind in cognitive science, I believe, is the notion of a virtual machine (VM) together with its associated notion of implementation.

Virtual Machines and the Philosophy of Mind

Virtual machines are quite common in computer science. Familiar examples of VMs include chess programs, word processors, the World Wide Web and e-mail systems, operating systems, interpreters, the SCHEME virtual machine, and so on. They are machines in that they are generally complex systems with an architecture, which specifies the number and kinds of interacting components and their contributions to the capabilities of the (overall) machine. Yet, they are not physical machines, whose components and interactions are partly defined in terms of physical objects and physical laws ("physical" understood in a wide sense). Rather, they are functionally specified *information-processing machines* that manipulate complex information structures and are implemented in terms of lower-level (physical) machines. What blueprints, circuit diagrams, and other ways of specifying the architecture of physical machines are to the engineer, programs are to the software engineer. They specify "components" (functions, data structures, etc.) and how they are related spatially (e.g., where they are placed in "logical space" or "virtual memory") as well as temporally (e.g., how they are created and can change over time) through the specification of processes that operate on them.

Although VMs are not any less *real* than physical machines, their main features are not physical properties and physical behaviors, although they are implemented in physical properties and behaviors. And the interacting components of a virtual machine do not interact (directly) via physical causes (such as forces or voltages) even though they are implemented in physical parts that do. Just think of a computer virus that crashes a word processing VM. Stating it the way I just did effectively claims that it is the virus that causes the VM to crash (rather than the sequence of changes in the strength of the electromagnetic field at the set of spatial locations, which instantiate the PC). Note that nothing "physical" is broken after the virus crashed the VM—the CPU is still executing commands as it finds them present in memory—yet the VM has ceased to exist (note that

turning off the power to the system has the same effect at the VM level, yet at the cost of a radical physical change).

VMs typically have their own set of concepts, which define and are defined for the realm (and level of abstraction/description) in/on which they work. Any description of a VM will make use of properties, functions, relations, and so on particular to that domain. A word processor, for example, manipulates words and paragraphs, though these cannot be found in the underlying physical machine. More specifically, the architecture of a word processing VM uses an ontology that includes entities like *letters, words, paragraphs,* properties like *italics, bold, double-spaced,* and relations like *in-between, first-on-page,* and *anchored-at.* Nowhere in the ontology of a word processing VM do we find *bits, bytes, registers, for-loops, stacks, arrays,* or *procedures*—rather they are part of the ontology of the "implementing" VM (which in turn is implemented in some lower-level physical machine). It should not come as a surprise that many VMs do not have a meaningful physical description (e.g., what would it mean to describe "section break" or "spell check" in terms of physics?). Nor will any decomposition of their components into smaller parts be meaningful.

Consequently, it is extremely hard if not impossible to understand whether and when a computer is implementing a word processing program if one is allowed to look only at the level lower and not that of the "word processing virtual machine" (e.g., the level of machine code or electric fields); if for nothing else, then for the unavoidable fact that there are indefinitely many virtual machines implemented by the lower-level system. Obviously, these problems are not restricted to the ontology and epistemology of VMs, but apply equally well to other domains (e.g., the realm of representations).

Another complicating factor in establishing the relation between physical systems and VMs is that physical systems might only "partially" implement VMs at any given time (e.g., because the VMs contain infinitely many components or components that are per se unbounded—note that VMs are not limited to computational abstractions such as Turing machines, but include uncomputable VMs specified by architectures such as neural networks with arbitrary real-numbered weight connections). The most conspicuous examples are standard PCs, which are considered to be "universal computers" (in the sense of the universal Turing machine),

except that they are universal only *counterfactually:* if they had enough memory, wide enough data registers and address busses, and so on, then they would be able to implement any computation. De facto, however, they are only finite state machines that can be "in some sense" described as implementing a universal Turing machine (which amounts to saying that they implement it only partially).

Although VMs get their causal powers from the physical and can affect things because they are (eventually) physically implemented, they may still be defined in terms of concepts different from those of their physical realizers. This raises a fundamental, open question: how is it that we can implement VMs whose ontology is different and possibly not reducible to that of the implementing system? (Obviously it can be done as we do have implementations of chess computers, word processors, etc.) I suspect that an answer to this challenge will not only be of great importance to computer science, but will also shed new light on the mind-body relation.

The Next Generation Again

There is a striking similarity between the relations among mental states, mental concepts, physical states, and physical concepts, on the one hand, and VM states, VM concepts, physical states, and physical concepts, on the other. One part of that similarity looks like a restatement of the well-known "computer metaphor"—mind is to matter as virtual machines are to matter—except that the left-hand side of the analogy will turn out to be an instance of the right-hand side if we view minds, like computations, as special kinds of virtual machines. This amounts to a syntactically minor but semantically significant modification of the defining statement of computationalism: *mental states are VM states* (leaving open in what sense and to what extent these VMs are computational in the classical sense).

The second part of the similarity points to an analogy between conceptual relationships: mental concepts are to physical concepts as virtual machine concepts are to physical concepts in at least the respect that neither mental nor virtual machines concepts need to be "reducible" to physical concepts in order to make sense of them and to be able to accommodate them in a physical world. The mental states defined by mental concepts

are as much part of this world if instantiated in physical states as are virtual machine states defined by virtual machines concepts.

More radically, I want to add the claim that (all/some?) *mental concepts are virtual machine concepts*. While some researchers are still after the "true" analysis of the concepts of folk psychology (hoping to cast them in terms of clearly specified classical concepts), and others still "believe" that mental talk should be banned from scientific discourse altogether, I want to suggest a middle-of-the-road approach. It concedes that mental concepts are cluster concepts as suggested by their stubborn resistance to any analysis in terms of classical concepts (with necessary and sufficient conditions). Hence there is no hope to achieve the former goal of analyzing the concepts of folk psychology. Yet, it questions both the claim that being cluster concepts necessarily renders them useless or unintelligible, and the conclusion that they consequently need to be eliminated.

Instead, this view attempts to make (at least some of) our intuitive, ordinary concepts more precise and useful by viewing them as implicitly architecture-based, that is, as being defined relative to and hence explanatorily dependent on a particular (cognitive) architecture. For example, the notion of "feeling guilty" can be explicated roughly as defining a class of cognitive processes that have the properties of frequently interrupting current processing by diverting attention to a particular past episode . . . It is the phrase "frequently interrupting current processing" that reveals an implicitly assumed architecture of parallel processes that allows some processes to interrupt others.

Once this architecture-based view is accepted, we can begin to explore the design options of architectures and different connections among architecture-based concepts, and hence mental concepts, bearing in mind that different architectures will support different families of architecure-based concepts, and therefore different mental concepts will be applicable to individuals with different architectures (e.g., insects are not capable of feeling guilt, because they lack the right kind of parallel, hierarchical, "reflective" architecture).

In sum, I want to suggest that computationalism should not be focused on viewing the mind as computational (in some fixed sense of "computational"), but rather be open to taking the wealth of resources and examples of virtual machines from computer science to address, study, and

answer deep philosophical, conceptual questions about cognition and the mind. I want to advocate a new view of computationalism that explicitly encompasses and acknowledges the realm of computing as a test bed for cognitive theories and as a playground for philosophical inquiry. This may include taking puzzling concepts from human psychology and transferring them to "computer psychology" to see if they could be applied and understood there, without any restrictions as to the kind of computing system. It may also include making philosophical questions precise and testing their answers by inventing and implementing concrete examples on computers.

I am convinced that in the future, more than ever, researchers in philosophy, psychology, and computer science will have to work together and join their forces in their efforts to explore the still unknown territory of the mind. Only then, will they (in Captain Kirk's words) be able to "continue the voyages we have begun, and journey to all those undiscovered countries, boldly going where no man . . . where no one . . . has gone before."

References

Agre, P. E. (1995). The soul gained and lost: Artificial intelligence as a philosophical project. *Stanford Humanities Review, 4(2),* 1–19.

Agre, P. E. (1997a). *Computation and Human Experience*. Cambridge: Cambridge University Press.

Agre, P. E. (1997b). Toward a critical technical practice: Lessons learned in trying to reform AI. In G. C. Bowker, S. L. Star, W. Turner, and L. Gasser (eds.), *Social Science, Technical Systems, and Cooperative Work: Beyond the Great Divide*. Hillsdale, NJ: Erlbaum.

Agre, P. E. and Chapman, D. (1987). Pengi: An implementation of a theory of activity. In *Proceedings of the Sixth National Conference on Artificial Intelligence, Seattle,* 268–222. San Mateo, CA: Morgan Kaufmann.

Allen, J., Hendler, J., and Tate, A., eds. (1990). *Readings in Planning*. Los Altos, CA: Morgan Kaufmann.

Andrews, J., Livingston, K., and Harnad, S. (1998). Categorical perception effects induced by category learning. *Journal of Experimental Psychology: Learning, Memory, and Cognition, 24(3),* 732–753.

Augarten, S. (1985). *Bit by Bit—An Illustrated History of Computers*. London: George Allen and Unwin.

Barwise, J. and Perry, J. (1983). *Situations and Attitudes*. Cambridge, MA: MIT Press.

Beaudoin, L. P. (1994). *Goal processing in autonomous agents*. Ph.D. thesis, School of Computer Science, The University of Birmingham.

Bechtel, W. and Richardson, R. C. (1993). *Discovering Complexity: Decomposition and Localization as Strategies in Scientific Research*. Princeton, NJ: Princeton University Press.

Block, N. (1978). Troubles with functionalism. In C. W. Savage (ed.), *Perception and Cognition: Issues in the Foundations of Psychology,* 261–325. Minneapolis: University of Minnesota Press.

Boolos, G. S. and Jeffrey, R. C. (1980) *Computability and Logic*, second edition. Cambridge: Cambridge University Press.

Bordo, B. (1987). *The Flight to Objectivity: Essays on Cartesianism and Culture*. Albany, NY: State University of New York Press.

Boulding, K. E. (1956). *The Image: Knowledge in Life and Society*. Ann Arbor, MI: University of Michigan Press.

Bridgeman, B. (1980). Brains + programs = minds. Reply to Searle. *Brain and Behavioral Sciences, 3,* 427–428.

Brooks, R. A. (1991). Intelligence without representation. *Artificial Intelligence Journal, 47,* 139–159.

Burge, T. (1979). Sinning against Frege. *Philosophical Review, 88(4),* 398–432.

Busemeyer, J. R. and Townsend, J. T. (1993). Decision field theory: A dynamic cognitive approach to decision making. *Psychological Review, 100,* 432–459.

Calude, C. S., Casti, J., and Dinneen, M. J., eds. (1998). *Unconventional Models of Computation*. Singapore: Springer-Verlag.

Cangelosi, A., Greco, A., and Harnad, S. (2000). From robotic toil to symbolic theft: Grounding transfer from entry-level to higher-level categories. *Connection Science, 12(2),* 143–162.

Cangelosi, A. and Harnad, S. (to appear). The adaptive advantage of symbolic theft over sensorimotor toil: Grounding language in perceptual categories. *Evolution of Communication* (Special Issue on Grounding).

Carpenter, B. E. and Doran, R. W., eds. (1986). *A. M. Turing's ACE Report of 1946 and Other Papers*. Cambridge, MA: MIT Press.

Chaitin, G. (1988). Randomness in arithmetic. *Scientific American, 259,* 80–85.

Chalmers, D. J. (1996). Does a rock implement every finite-state automation? *Synthese, 108,* 310–333.

Chomsky, N. (1965). *Aspects of the Theory of Syntax*. Cambridge, MA: MIT Press.

Church, A. (1936a). A note on the *Entscheidungsproblem*. *Journal of Symbolic Logic, 1,* 40–41.

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

Church, A. (1937). Review of Turing 1936. *Journal of Symbolic Logic, 2,* 42–43.

Church, A. (1940). On the concept of a random sequence. *American Mathematical Society Bulletin, 46,* 130–135.

Churchland, P. M. and Churchland, P. S. (1983). Stalking the wild epistemic engine. *Nous, 17,* 5–18.

Churchland, P. M. and Churchland, P. S. (1990). Could a machine think? *Scientific American, 262,* 26–31.

Cohen, P., Greenberg, M. L., Hart, D. M., and Howe, A. E. (1989). Trial by fire: Understanding the design requirements for agents in complex environments. *AI Magazine, 10(3),* 32–48.

Cooper, R. and Shallice, T. (2000). Contention scheduling and the control of routine activities. *Cognitive Neuropsychology, 17(4),* 297–338.

Copeland, B. J. (1993). *Artificial Intelligence: A Philosophical Introduction.* Oxford: Blackwell.

Copeland, B. J. (1994). Artificial intelligence. In Guttenplan (1994), 122–131.

Copeland, B. J. (1996). What is computation? *Synthese, 108,* 335–359.

Copeland, B. J. (1997). The broad conception of computation. *American Behavioral Scientist, 40,* 690–716.

Copeland, B. J. (1998a). A lecture and two radio broadcasts on machine intelligence by Alan Turing. *Machine Intelligence, 15,* 445–475.

Copeland, B. J. (1998b). Even Turing machines can compute uncomputable functions. In Calude, Casti, and Dinneen (1998), 150–164.

Copeland, B. J. (1998c). Super Turing machines. *Complexity, 4* (October), 30–32.

Copeland, B. J. (1998d). Turing's O-machines, Penrose, Searle, and the brain. *Analysis, 58,* 128–138.

Copeland, B. J. and Proudfoot, D. (1999). Alan Turing's forgotten ideas in computer science. *Scientific American, 280,* 99–103.

Copeland, B. J. and Sylvan, R. (1999). Beyond the universal Turing machine. *Australasian Journal of Philosophy* (March) *273,* 46–66.

Cottingham, J., Stoothoff, R., and Murdoch, D. (1984–1985). *The Philosophical Writings of Descartes,* vols. I and II. Cambridge: Cambridge University Press.

Cottingham, J., Stoothoff, R., Murdoch, D., and Kenny, A. (1991). *The Philosophical Writings of Descartes,* vol. III. Cambridge: Cambridge University Press.

Crevier, D. (1993). *AI: The Tumultuous History of the Search for Artificial Intelligence.* New York: Basic Books.

Curry, H. B. (1929). An analysis of logical substitution. *American Journal of Mathematics, 51,* 363–384.

Curry, H. B. (1930). Grundlagen der kombinatorischen Logik. *American Journal of Mathematics, 52,* 509–536, 789–834.

Curry, H. B. (1932). Some additions to the theory of combinators. *American Journal of Mathematics, 54,* 551–558.

Damper, R. I. and Harnad, S. (2000). Neural network modeling of categorical perception. *Perception and Psychophysics, 62(4),* 843–867.

Davis, M. (ed.) (1965). *The Undecidable.* New York: Raven Press.

Dennett, D. C. (1978). *Brainstorms: Philosophical Essays on Mind and Psychology.* Cambridge, MA: MIT Press.

Dennett, D. C. (1991). *Consciousness Explained.* Boston: Little, Brown and Company.

Dennett, D. C. (1995). *Darwin's Dangerous Idea: Evolution and the Meanings of Life.* New York: Simon and Schuster.

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

Devitt, M. (1991). Why Fodor can't have it both ways. In Loewer, B. and Rey, G. (eds.), *Meaning in Mind: Fodor and His Critics.* Oxford: Basil Blackwell, 95–118.

Dietrich, E. (1990). Computationalism. *Social Epistemology, 4(2),* 135–154.

Dretske, F. I. (1981). *Knowledge and the Flow of Information.* Cambridge, MA: MIT Press.

Dreyfus, H. L. (1979). *What Computers Can't Do: A Critique of Artificial Reason,* revised edition. New York: Harper and Row.

Dreyfus, H. L. (1992). *What Computers Still Can't Do: A Critique of Artificial Reason.* Cambridge, MA: MIT Press.

Feigenbaum, E. A. and Feldman, J., eds. (1963). *Computers and Thought.* New York: McGraw-Hill.

Fikes, R. E., Hart, P. E., and Nilsson, N. J. (1972). Learning and executing generalized robot plans. *Artificial Intelligence, 3(4),* 251–288.

Firby, R. J. (1987). An investigation into reactive planning in complex domains. In *Proceedings of the Sixth National Conference on Artificial Intelligence, Seattle,* 202–206. San Mateo, CA: Morgan Kaufmann.

Fodor, Jerry A. (1975). *The Language of Thought.* Cambridge, MA: Harvard University Press.

Fodor, J. A. (1980). Methodological solipsism considered as a research strategy in cognitive psychology. *Behavioral and Brain Sciences, 3(1),* 63–73. Reprinted in Fodor (1981a).

Fodor, J. A. (1981a). *RePresentations: Philosophical Essays on the Foundations of Cognitive Science.* Cambridge, MA: MIT Press.

Fodor, J. A. (1981b). The mind-body problem. *Scientific American, 244,* 124–32.

Fodor, J. A. (1983). *The Modularity of Mind.* Cambridge, MA: MIT Press.

Fodor, J. A. (1984). Semantics, Wisconsin style. *Synthese, 59,* 231–250.

Fodor, J. A. (1987). *Psychosemantics: The Problem of Meaning in the Philosophy of Mind.* Cambridge, MA: MIT Press.

Fodor, J. A. (1994). *The Elm and the Expert: Mentalese and Its Semantics.* Cambridge, MA: MIT Press.

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

Fox, M. S. and Smith, S. (1984). ISIS: A knowledge-based system for factory scheduling. *Expert Systems, 1(1),* 25–49.

Frege, G. (1960). In P. Geach and M. Black (eds.), *Translations from the Philosophical Writings.* Oxford: Blackwell.

Gandy, R. (1980). Church's Thesis and principles for mechanism. In J. Barwise, H. J. Keisler, and K. Kunen (eds.), *Proceedings of the Kleene Symposium*. Amsterdam: North-Holland Publishing Company.

Gandy, R. (1988). The confluence of ideas in 1936. In Herken (1988).

Gardner, H. (1985). *The Mind's New Science: A History of the Cognitive Revolution*. New York: Basic Books Publishers.

Gaukroger, S. (1995). *Descartes: An Intellectual Biography*. Oxford: Clarendon Press.

Gaukroger, S. (1997). *The Genealogy of Knowledge*. Manuscript.

Georgeff, M. P. (1987). Planning. In J. F. Traub, B. J. Grosz, B. W. Lampson, and N. J. Nilsson (eds.), *Annual Review of Computer Science 2*. Palo Alto, CA: Annual Reviews Inc., 359–400.

Gerhard, C. J. (1875–90). *Die philosophischen Schriften von Leibniz*. 7 vols. Berlin: Weisman. Reprinted (1965), Hildesheim: Olms.

Geroch, R. and Hartle, J. B. (1986). Computability and physical theories. *Foundations of Physics, 16,* 533–550.

Gödel, K. (1936). Über die Lange von Beweisen. *Ergebnisse eines mathematischen Kolloquiums, 7,* 23–24.

Gomes, C. P., Selman, B., and Kautz, H. (1998). Boosting combinatorial search through randomization. *Proceedings of the Fifteenth National Conference on Artificial Intelligence,* Madison, WI. AAAI Press/The MIT Press.

Goodwin, C. (1981). *Conversational Organization: Interaction between Speakers and Hearers*. New York: Academic Press.

Gregory, R. L. (1987). *The Oxford Companion to the Mind*. Oxford: Oxford University Press.

Guttenplan, S. (1994). *A Companion to the Philosophy of Mind*. Oxford: Blackwell.

Hallam, J. C. T. and Malcolm, C. A. (1994). Behavior—perception, action, and intelligence—The view from situated robotics. In *Philosophical Transactions of the Royal Society A, 349, (1689),* 29–42.

Halpern, J. (1987). Using reasoning about knowledge to analyze distributed systems. *Annual Review of Computer Science, 2,* 37–68.

Harnad, S. (1976). Induction, evolution, and accountability. *Annals of the New York Academy of Sciences, 280,* 58–60.

Harnad, S. (1987). The induction and representation of categories. In Harnad, S. (ed.), *Categorical Perception: The Groundwork of Cognition*. New York: Cambridge University Press.

Harnad, S. (1990). The symbol grounding problem. *Physica, D 42,* 335–346.

Harnad, S. (1993a). Grounding symbols in the analog world with neural nets. *Think 2(1),* 12–78. Special issue on *Connectionism versus Symbolism,* Powers, D. M. W. and Flach, P. A. (eds.).

Harnad, S. (1993b). Problems, problems: The frame problem as a symptom of the symbol grounding problem. *PSYCOLOQUY 4(34).*

Harnad, S. (1994). Computation is just interpretable symbol manipulation: Cognition isn't. *Minds and Machines, 4,* 379–390. Special Issue on *What Is Computation.*

Harnad, S. (1996a). Experimental analysis of naming behavior cannot explain naming capacity. *Journal of the Experimental Analysis of Behavior, 65,* 262–264.

Harnad, S. (1996b). The origin of words: A psychophysical hypothesis. In B. Velichkovsky and D. Rumbaugh (eds.), *Communicating Meaning: Evolution and Development of Language.* Mahwah, NJ: Erlbaum, 27–44.

Harnad, S. (2000a). Correlation vs. Causality: How/why the mind/body problem is hard. (Invited Commentary of Humphrey, N., *How to Solve the Mind-Body Problem.*) *Journal of Consciousness Studies, 7(4),* 54–61.

Harnad, S. (2000b). Minds, machines, and Turing: The indistinguishability of indistinguishables. *Journal of Logic, Language, and Information, 9(4),* 425–445.

Harnad, S. (2001a). No easy way out. *The Sciences, 41(2),* 36–42.

Harnad, S. (2001b). What's wrong and right about Searle's Chinese room argument? In M. Bishop and J. Preston (eds.), *Essays on Searle's Chinese Room Argument.* Oxford: Oxford University Press.

Harnad, S., Hanson, S. J., and Lubin, J. (1991). Categorical perception and the evolution of supervised learning in neural nets. In D. W. Powers and L. Reeker (eds.), *Proceedings of the AAAI Spring Symposium on Machine Learning of Natural Language and Ontology,* Document D91–09, Deutsches Forschungszentrum fur Kuenstliche Intelligenz GmbH Kaiserslautern FRG, 65–74.

Haugeland, J. (1978). The nature and plausibility of cognitivism. *Behavioral and Brain Sciences, 1,* 215–226.

Haugeland, J. (1981a). Semantic engines. In Haugeland (1981b), pp. 1–34.

Haugeland, J., ed. (1981b). *Mind Design I.* Cambridge, MA: MIT Press.

Haugeland, J. (1982). Analog and analog. *Philosophical Topics* (spring 1981); reprinted in J. I. Biro and Robert W. Shahan (eds.), *Mind, Brain, and Function: Essays in the Philosophy of Mind.* Norman, OK: University of Oklahoma Press, 213–225.

Haugeland, J. (1985). *Artificial Intelligence: The Very Idea.* Cambridge, MA: MIT Press.

Haugeland, J., ed. (1996a). *Mind Design II.* Cambridge, MA: MIT Press.

Haugeland, J. (1996b). What is mind design? In J. Haugeland (1996a).

Hawking, Stephen W. (1988). *A Brief History of Time.* Toronto: Bantam Books.

Held, R. and Hein, A. (1963). Movement-produced stimulation in the development of visually guided behavior. *Journal of Comparative and Physiological Psychology, 56(5),* 872–876.

Hendler, J., ed. (1990). Planning in uncertain, unpredictable, or changing environments. In *Proceedings of the AAAI Symposium at Stanford,* University of Maryland Systems Research Center Report SRC TR 90-45.

Henry, G. C. (1993). *The Mechanism and Freedom of Logic.* Lanham, MD: University Press of America.

Herken, R., ed. (1988). *The Universal Turing Machine: A Half-Century Survey.* Berlin: Kammerer and Unverzagt.

Hilbert, D. and Ackermann, W. (1928). *Grundzüge der Theoretischen Logik.* Berlin: Springer.

Hobbes, T. (1651). *Leviathan.* London: Andrew Crooke. Reprinted by Scolar Press, Menston, 1969.

Hobbes, T. (1994). *Leviathan.* In *The Collected Works of Thomas Hobbes.* Routledge.

Hodges, A. (1988). Alan Turing and the Turing machine. In Herken (1988).

Hodges, A. (1992). *Alan Turing: The Enigma.* London: Vintage.

Hodges, A. (1997). *Turing.* London: Phoenix.

Hogarth, M. L. (1994). Non–Turing computers and non–Turing computability. *PSA 1994, vol. 1,* 126–138.

Hopcroft, J. E. and Ullman, J. D. (1979). *Introduction to Automata Theory, Languages, and Computation.* Reading, MA: Addison-Wesley.

Hurlbert, A. and Poggio, T. (1988). Making machines (and artificial intelligence) see. *Daedalus, 117(1),* 213–239.

Hutchins, E. (1995). *Cognition in the Wild.* Cambridge, MA: MIT Press.

Johnson-Laird, P. N. (1987). How could consciousness arise from the computations of the brain? In C. Blakemore and S. Greenfield (eds.), *Mindwaves.* Oxford: Basil Blackwell.

Johnson-Laird, P. N. (1988). *The Computer and the Mind.* Cambridge, MA: Harvard University Press.

Kant, I. (1781). *Critique of Pure Reason.* Translated (1929) by Norman Kemp Smith. London: Macmillan.

Kennedy, C. M. (1999). Distributed reflective architectures for adjustable autonomy. *International Joint Conference on Artificial Intelligence (IJCAI99).* Paper given at Workshop on Adjustable Autonomy, Stockholm, Sweden.

King, D. (1996). Is the human mind a Turing machine? *Synthese, 108,* 379–389.

Kleene, S. C. (1952). *Introduction to Metamathematics.* Amsterdam: North-Holland.

Kleene, S. C. (1967). *Mathematical Logic.* New York: Wiley.

Knoespel, K. J. (1987). The narrative matter of mathematics: John Dee's preface to the *Elements* of Euclid of Megara (1570). *Philological Quarterly, 66(1),* 27–46.

Lakoff, G. and Johnson, M. (1980). *Metaphors We Live By.* Chicago: University of Chicago Press.

Langton, C. R. (1989). Artificial life. In Langton (ed.), *Artificial Life,* 1–47. Redwood City, CA: Addison-Wesley.

Lashley, K. S. (1951). The problem of serial order in behavior. In Lloyd A. Jeffress (ed.), *Cerebral Mechanisms in Behavior: The Hixon Symposium.* New York: Wiley.

Leibniz, G. W. (1875–90). *De Arte Combinatoria,* in *Die philosophischen Schriften von Leibniz.* 7 vols. Berlin: Weisman.

Lucas, J. R. (1961). Minds, machines, and Gödel. *Philosophy, 36,* 122–127.

Lucas, J. R. (1996). Minds, machines, and Gödel: A retrospect. In P. Millican and A. Clark (eds.), *Machines and Thought.* Oxford: Clarendon Press.

Markov, A. A. (1960). The theory of algorithms. *American Mathematical Society Translations, series 2, 15,* 1–14.

Marr, D. (1982). *Vision.* San Francisco: Freeman.

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

McGuire, J. E. (1972). Boyle's conception of nature. *Journal of the History of Ideas, 33,* 523–542.

Melnyk, A. (1996). Searle's abstract argument against strong AI. *Synthese, 108,* 391–419.

Mendelson, E. (1964). *Introduction to Mathematical Logic.* New York: Van Nostrand.

Miller, G. A., Galanter, E., and Pribram, K. H. (1960). *Plans and the Structure of Behavior.* New York: Henry Holt.

Millikan, R. (1984). *Language, Thought, and Other Biological Categories.* Cambridge, MA: MIT Press.

Millikan, R. (1989). Biosemantics. *Journal of Philosophy, 86, 6,* 281–297.

Millikan, R. (1990). Compare and contrast Dretske, Fodor, and Millikan on teleosemantics. *Philosophical Topics, 18, 2,* 151–161.

Minsky, M. L. (1961). Steps towards artificial intelligence. In E. A. Feigenbaum and J. Feldman (1963).

Molesworth, W. (1994). *The Collected Works of Thomas Hobbes.* Routledge.

Newell, A. (1980). Physical symbol systems. *Cognitive Science, 4,* 135–183.

Newell, A., Shaw, J. C., and Simon, H. A. (1960). Report on a general problem-solving program. In *Proceedings of the International Conference on Information Processing, Paris,* 256–264. UNESCO.

Newell, A. and Simon, H. A. (1963). GPS, A program that simulates human thought. In E. A. Feigenbaum and J. Feldman (1963), 279–293 (Originally published in 1961).

Newell, A. and Simon, H. A. (1976). Computer science as empirical inquiry: Symbols and search. *Communications of the Association for Computing Machinery, 19(3),* 113–126.

Odifreddi, P. (1989). *Classical Recursion Theory.* Amsterdam: North-Holland.

Pain, S. (2000). Liquid assets. *New Scientist, 2268,* 46–47.

Payton, D. W., Rosenblatt, J. K., and Keirsey, D. M. (1990). Plan guided reaction. *IEEE Transactions on Systems, Man, and Cybernetics, 20(6),* 1370–1382.

Penrose, R. (1989). *The Emperor's New Mind: Concerning Computers, Minds and the Laws of Physics,* Oxford: Oxford University Press.

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

Pevtzow, R. and Harnad, S. (1997). Warping similarity space in category learning by human subjects: The role of task difficulty. In M. Ramscar, U. Hahn, E. Cambouropolos, and H. Pain (eds.), *Proceedings of SimCat 1997: Interdisciplinary Workshop on Similarity and Categorization.* Department of Artificial Intelligence, Edinburgh University, 189–195.

Port, R. and van Gelder, T., eds. (1995). *Mind as Motion: Explorations in the Dynamics of Cognition.* Cambridge, MA: MIT Press.

Post, E. L. (1943). Formal reductions of the general combinatorial decision problem. *American Journal of Mathematics, 65,* 197–215.

Post, E. L. (1946). A variant of a recursively unsolvable problem. *Bulletin of the American Mathematical Society, 52,* 264–268.

Pratt, V. (1987). *Thinking Machines—The Evolution of Artificial Intelligence.* Oxford: Basil Blackwell.

Putnam, H. (1975). The meaning of meaning. In: *Mind, Language and Reality: Philosophical Papers, Volume 2.* Cambridge: Cambridge University Press.

Putnam, H. (1988). *Representation and Reality.* Cambridge, MA: MIT Press.

Putnam, H. (1992). *Renewing Philosophy.* Cambridge, MA: Harvard University Press.

Pylyshyn, Z. W. (1984). *Computation and Cognition: Toward a Foundation for Cognitive Science.* Cambridge, MA: MIT Press.

Pylyshyn, Z. W., ed. (1987). *The Robot's Dilemma: The Frame Problem in Artificial Intelligence.* Norwood, NJ: Ablex.

Ryle, G. (1949). *The Concept of Mind.* London: Hutchinson.

Scheutz, M. (1999). When physical systems realize functions. . . . *Minds and Machines, 9, 2,* 161–196.

Scheutz, M., ed. (2000). *New Computationalism. Conceptus Studien 14.* St. Augustin: Academia Verlag.

Scheutz, M. (2001). Computational versus causal complexity. *Minds and Machines 11,* 543–566.

Schönfinkel, M. (1924). Über die Bausteine der mathematischen. *Mathematische Annalen, 92,* 305–316.

Searle, J. (1980). Minds, brains, and programs. *The Behavioral and Brain Sciences, 3(3),* 417–424.

Searle, J. (1990). Is the brain's mind a computer program? *Scientific American, 262,* 20–25.

Searle, J. (1992). *The Rediscovery of the Mind.* Cambridge, MA: MIT Press.

Searle, J. (1997). *The Mystery of Consciousness.* New York: New York Review.

Shagrir, O. (1997). Two dogmas of computationalism. *Minds and Machines, 7,* 321–344.

Shannon, C. E. and Weaver, W. (1949). *The Mathematical Theory of Communication.* Urbana, Illinois: The University of Illinois Press.

Shepherdson, J. C. and Sturgis, H. E. (1963). Computability of recursive functions. *Journal of the Association for Computing Machinery, 10,* 217–255.

Sloman, A. (1978). *The Computer Revolution in Philosophy.* Hassocks, Sussex: Harvester Press and Humanities Press. (Out of print. See online version at http://www.cs.bham.ac.uk/research/cogaff/crp/)

Sloman, A. (1985). What enables a machine to understand? In Aravind K. Joshi (ed.), *Proceedings of 9th IJCAI, Los Angeles,* 995–1001. Morgan Kaufmann.

Sloman, A. (1987). Reference without causal links. In J. B. H. du Boulay, D. Hogg, and L. Steels (eds.), *Advances in Artificial Intelligence II,* North-Holland, Dordrecht, 369–381.

Sloman, A. (1992). The emperor's real mind. Review of Roger Penrose's *The Emperor's New Mind: Concerning Computers, Minds and the Laws of Physics, Artificial Intelligence, 56,* 355–396.

Sloman, A. (1997). What sort of control system is able to have a personality? In R. Trappl and P. Petta (eds.), *Creating Personalities for Synthetic Actors: Towards Autonomous Personality Agents, Lecture Notes in AI.* Berlin: Springer, 166–208.

Sloman, A. (1998). Damasio, Descartes, alarms and meta-management. In *Proceedings International Conference on Systems, Man, and Cybernetics (SMC98).* IEEE: San Diego, 2652–2657.

Sloman, A. (1999). What sort of architecture is required for a human-like agent? In M. Wooldridge and A. Rao (eds.), *Foundations of Rational Agency.* Dordrecht: Kluwer Academic, 35–52.

Sloman, A. (2000a). Architectural requirements for human-like agents both natural and artificial. (What sorts of machines can love?) In K. Dautenhahn (ed.), *Human Cognition and Social Agent Technology: Advances in Consciousness Research.* Amsterdam: John Benjamins, 163–195.

Sloman, A. (2000b). Interacting trajectories in design space and niche space: A philosopher speculates about evolution. In M. Schoenauer, K. Deb, G. Rudolph, X. Yao, E. Lutton, J. J. Merelo, and H. P. Schwefel (eds.), *Parallel Problem Solving from Nature, PPSN VI, Lecture Notes in Computer Science, no. 1917,* Springer-Verlag, 3–16.

Sloman, A. (2001a). Beyond shallow models of emotion. *Cognitive Processing International Quarterly of Cognitive Science, 2, 1,* 177–198.

Sloman, A. (2001b). Diagrams in the mind. In M. Anderson, B. Meyer, and P. Olivier (eds.), *Diagrammatic Representation and Reasoning*. Springer-Verlag, Berlin.

Sloman, A. (to appear). How many separately evolved emotional beasties live within us? In R. Trappl and P. Petta (eds.), *Emotions in Humans and Artifacts*. Cambridge, MA: MIT Press.

Sloman, A. and Logan, B. (2000). Evolvable architectures for human-like minds. In G. Hatano, N. Okada, and H. Tanabe (eds.), *Affective Minds*. Elsevier, Amsterdam, 169–181.

Smith, B. C. (1996). *The Origin of Objects*. Cambridge, MA: MIT Press.

Smith, B. C. (forthcoming). *The Age of Significance*. 7 vols. Cambridge, MA: MIT Press.

Smith, B. C. (in press). Reply to Dan Dennett. In Hugh Clapin (ed.), *Philosophers of Mental Representation*. Oxford University Press (forthcoming 2002).

Smith, L. B. and Thelen, E. (1994). *A Dynamic Systems Approach to the Development of Cognition and Action*. Cambridge, MA: MIT Press.

Smolensky, P. (1988). On the proper treatment of connectionism. *Behavioral and Brain Sciences, 11,* 1–23.

Steels, L. (2001). Languages for autonomous robots. *IEEE Intelligent Systems, 16(6),* 16–22.

Steels, L. and Kaplan, F. (1999). Bootstrapping grounded word semantics. In T. Briscoe (ed.), *Linguistic Evolution through Language Acquisition: Formal and Computational Models*. Cambridge: Cambridge University Press.

Suchman, L. A. and Trigg, R. H. (1993). Artificial intelligence as craftwork. In S. Chaiklin and J. Lave (eds.), *Understanding Practice: Perspectives on Activity and Context*. Cambridge: Cambridge University Press.

Thomson, A., ed. (1996). *La Mettrie: Machine Man and Other Writings*. Cambridge: Cambridge University Press.

Tijsseling, A. and Harnad, S. (1997). Warping similarity space in category learning by backprop nets. In M. Ramscar, U. Hahn, E. Cambouropolos, and H. Pain (eds.), *Proceedings of SimCat 1997: Interdisciplinary Workshop on Similarity and Categorization*. Department of Artificial Intelligence, Edinburgh University, 263–269.

Turing, A. M. (1936). On computable numbers, with an application to the *Entscheidungsproblem*. *Proceedings of the London Mathematical Society, Series 2, 42,* 230–265.

Turing, A. M. (1939). Systems of logic based on ordinals. *Proceedings of the London Mathematical Society, 45,* 161–228.

Turing, A. M. (1945). Proposal for development in the mathematics division of an automatic computing engine (ACE). In B. E. Carpenter and R. W. Doran (1986).

Turing, A. M. (1947). Lecture to the London Mathematical Society on 20 February 1947. In B. E. Carpenter and R. W. Doran (1986).

Turing, A. M. (1948). Intelligent machinery. National Physical Laboratory report. In B. Meltzer and D. Michie (eds.), *Machine Intelligence 5*. Edinburgh: Edinburgh University Press, 1969.

Turing, A. M. (1949). *Generation of Random Numbers*. Manchester: University of Manchester Computing Laboratory.

Turing, A. M. (1950a). Computing machinery and intelligence. *Mind, 59,* 433–460. Reprinted in E. A. Feigenbaum and J. Feldman (1963), 11–35.

Turing, A. M. (1950b). *Programmers' Handbook for Manchester Electronic Computer*. Manchester: University of Manchester Computing Laboratory.

Turing, A. M. (1951a). Intelligent machinery, a heretical theory. In Copeland (1998b).

Turing, A. M. (1951b). Can digital computers think? BBC Radio Broadcast. In Copeland (1998b).

Turing, A. M. et al. (1952). Can automatic calculating machines be said to think? In Copeland (1998b).

van Gelder, T. (1995). What might cognition be, if not computation? *Journal of Philosophy, 91,* 345–381.

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

Vera, A. H. and Simon, H. A. (1993). Situated action: A symbolic interpretation. *Cognitive Science, 17,* 7–48.

Walkerdine, V. (1988). *The Mastery of Reason: Cognitive Development and the Production of Rationality*. London: Routledge.

Weinberg, S. (1994). *Dreams of a Final Theory*. Vintage Books.

Williams, M. R. (1997). *A History of Computing Technology*, second ed. Los Alamitos, CA: IEEE Computer Society Press.

Wittgenstein, L. (1953). *Philosophical Investigations*. Translated by G. E. M. Anscombe, third ed., Oxford: Blackwell, 1967.

Wittgenstein, L. (1980). *Remarks on the Philosophy of Psychology*, vol. 1. Oxford: Blackwell.

Woolgar, S. (1991). Configuring the user: The case of usability trials. In John Law (ed.), *A Sociology of Monsters: Essays on Power, Technology and Domination*. London: Routledge.

Wright, I. P., Sloman, A., and Beaudoin, L. P. (1996). Towards a design-based analysis of emotional episodes. *Philosophy, Psychiatry, and Psychology, 3, 2,* 101–126.

Index