From Matthias Scheutz, ed., *Computationalism: New Directions*, MIT Press, 2002, pp. 23–58.

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 processing (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 observation 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