Four Dialectics of Computing

Brian Cantwell Smith*

Four dialectics underwrite all of computing:

1. Between **meaning** and **mechanism**;
2. Between the **abstract** and the **concrete**;
3. Between the **static** and the **dynamic**; and
4. Between what is **singular** (one) and what is **plural** (many).

Any adequate account of computing will, among other things, have to do justice to all four. In fact the quartet can be used as something of a normative standard, in terms of which to evaluate any proposed theory.

I want to describe these dialectics in some detail—what they come to, how they relate, etc. But I will get to that only in Part II. First I want to report on a project I've been involved in where these four have arisen as central concerns. The results of that study alter the ways these dialectics should be approached.

Part I — Theory of Computing

1. Three criteria

For more than 30 years I have been searching for a "comprehensive theory of computing." Since the outset, I have taken that to be one that meets three criteria:

1. **Empirical:** It must do justice to computational practice (e.g., be capable of explaining Microsoft Word: the program, its construction, interpretation, maintenance, and grounds for use);

---

*Cognitive Science, Computer Science, Philosophy, and History and Philosophy of Science
Indiana University, Bloomington, IN 47405 USE
smithbc@indiana.edu
http://www.ageofsig.org/people/bcsmith

Version of May 10, 1999
Comments welcome

Note: This paper is extracted from, and consists in part of excerpts from, *Volume I (Introduce)* of *The Age of Significance: An Essay on the Foundations of Computation and Intentionality*, forthcoming from the MIT Press.

2. **Conceptual:** It must discharge all intellectual debts (e.g., to semantics), so that we can understand what it says, where it comes from, what it "costs"; and
3. **Cognitive:** It must provide a tenable foundation for the computational theory of mind—the thesis, sometimes known as "cognitivism," that underlies artificial intelligence and cognitive science.

All three criteria are intended to be sensible. The first, empirical, one, of doing justice to practice, helps to keep the analysis grounded in real-world examples. It is humbling, too, since computer technology so rapidly expands, dodges expectations, and in general outstrips our theoretical grasp. But the criterion's primary advantage is to give us a vantage point from which to question all theoretical perspectives—including (perhaps especially) the widely-held Turing-machine or recursion-theoretic conception of computability that currently claims the title "the theory of computation." For I take it as a tenet that what Silicon Valley *treats* as computational *is* computational; to deny that will be considered sufficient grounds for rejection. But I am not prepared to accord such a priori commitment to any *story* about computation—including the theory we currently teach in graduate school. I also reject all proposals that assume that computation can be *defined*. By my lights—and in spite of the peculiar fact that we construct the evidence—computer science is primarily an empirical discipline. To make the grade, a theory must make substantive, empirical claims on what I call **computation in the wild**:[1] that eruptive body of practices, techniques, networks, machines, and behavior that has so palpably revolutionized late twentieth century life.

The second, "conceptual" criterion, that a theory own up to—and as far as possible discharge—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" already mentioned, as well as many reigning but largely tacit ideas about computing held in surrounding disciplines,[2] implicitly rely, without explana-

[1]Borrowed from Hutchins' *Cognition in the Wild* (1995).

[2]A notable example of such a far-from-innocent assumption is the common idea that "computation" is the fundamental notion, with a "computer" simply being any physical device

tion, on such substantial notions as representation, semantics, and interpretation.[3] Second, in spite of this reliance, there is a tendency throughout surrounding intellectual terrain to point to computation as a possible *theory of those very recalcitrant notions* (for example, to invoke notions of computability in analyses of language, meaning, and learning). Unless we ferret out all such dependencies, and lay them in plain view, we run the risk of endorsing accounts that are either based on, or give rise to, vicious circularity.

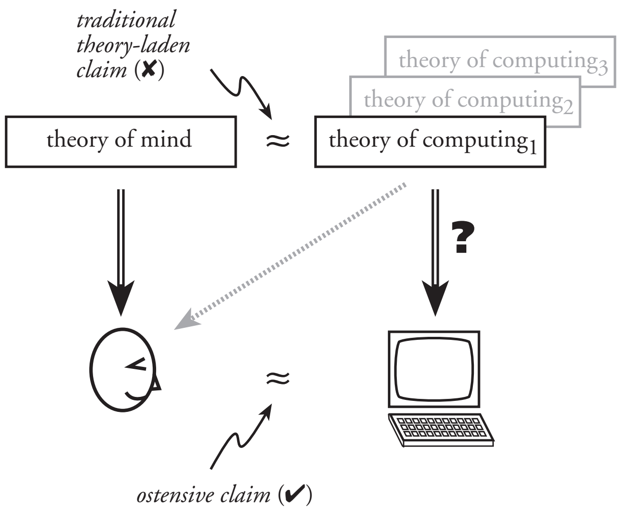

Figure 1 — The Computational Theory of Mind

The third "cognitive" condition, that an adequate theory of computation must provide a tenable foundation for a theory of mind, is of a somewhat different character. Like the second, it is more a metatheoretic requirement on the form or status of the theory than a constraint on substantive content. In endorsing the criterion, however, I make no commitment to cognitivism's being true or false. I just want to know what it says.

That is not to say that cognitivism's content is left entirely open. Its fundamental thesis—that the mind is computational—is given substance by the first, empirical criterion. The situation is depicted in figure 1. As I read it, that is, cognitivism is not a theory-laden proposal, in the sense of framing specific hypotheses about what computers are. Rather, it has more an ostensive character: that people (i.e., us) are computers in whatever way computers

that carries out a computation. It turns out, on inspection, that this assumption builds in a residually dualist stance towards the mind/body problem—something I eventually want to argue against, and probably not a claim that anyone should want to build into their theories as a presumptive but inexplicit premise.

[3]The official theory of computation is riddled with undischarged representational and modelling assumptions.

(i.e., those things over there) are computers—or at least in whatever way *some* of those things are computers. To know what cognitivism comes to, therefore, we need to know what computers are.[4]

An example will make this clear. Many in the philosophy of mind (such as Fodor) take the computational theory of mind (CCOM) to be a claim that cognition is formal symbol manipulation. On my view, to say that is to make two claims at once, that must be independently assessed: (i) that the mind is computational, and (ii) that to be computational is to be formal symbol manipulation. I am not sure about the former; but (as I will explain in a moment) I believe the latter is false.

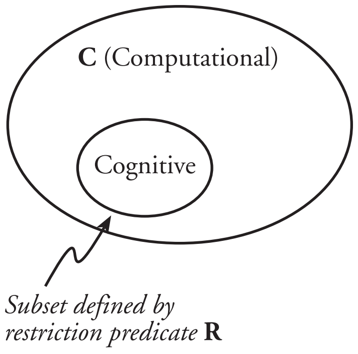

Figure 2 — Cognition as subset

Even once we separate an (ostensive) computational theory of mind from substantive empirical theories of what it is to be computational, we have not gone far enough. For few think that cognition and computation are the same thing (i.e., that every computer is a cognitive agent, that the terms 'cognition' and 'computation' should be taken as a posteriori synonymous). Rather, the idea, I take it, is that only *some* computers are cognitive: that cognition is something like a proper computational species. One can understand this by seeing that the claim involves computation at two levels (as depicted in figure 2). Within the presumptively (wider) class of "things

[4]It follows that any theoretical formulation of cognitivism is doubly contingent. Thus consider Newell and Simon's (1976) popular "physical symbol system hypothesis," according to which human intelligence is claimed to consist of physical symbol manipulation, or Fodor's (1975, 1980) claim that thinking consists of formal symbol manipulation, or Dreyfus' (1993) assertion that cognitivism (as opposed to connectionism) requires the explicit manipulation of explicit symbols. Not only do these writers make a hypothetical statement about *people*, that they are physical, formal, or explicit symbol manipulators, respectively; they do so by making a hypothetical statement about *computers*, that they are in some essential or illuminating way characterisable in the same way. Because I take the latter claim to be as subservient to empirical adequacy as the former (by the first criterion), there are two ways in which these writers could be wrong. In claiming that people are formal symbol manipulators, for example, Fodor would be wrong if computers were formal symbol manipulators and people were not. But he would also be wrong, *even though cognitivism itself might still be true*, if computers were not formal symbol manipulators, either.

computational" (**C**), the cognitive is not only claimed to be a subset, but the restrictive predicate (or characteristic function) **R** identifying that subset is *also assumed to be formulable in computational terms*. For example, suppose (falsely!) that what it is to be cognitive is to be a finite-state automaton with bounded input tape and look-ahead buffer of 3. To say this is to add to a background claim that cognitive systems are computers a more specific claim that the restrictive predicate identifying the cognitive subset is computationally-formulable (namely: finite-state automata with bounded tape and look-ahead buffer of 3).[5]

There is much more to be said about the relation between cognitivism and computing, but we need not go into it here. The basic point is merely to show that understanding the computational theory of mind depends on having an adequate account of computing. And so, to bring this back to the project being described, while my ultimate interest has always had primarily to do with people and cognitive science, my efforts for these 30-odd years have focused almost exclusively on the prior question: of what computing is—of what it is to be computational.[6]

2. Six construals

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

With regards to 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. I have found it convenient to distinguish six main *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

[5]Obviously one can identify subsets of **C** in the diagram that are not identified in computational terms: to be a computer that costs less than $1000, for example, or to be a computer that weighs exactly 7.3 lbs, are presumably non-computationally restricted subsets of the computational.

[6]Some will want to distinguish computer, computational, etc., in this phrasing. I agree, but making appropriate distinctions between such notions is exactly one of the tasks that a theory of computation should give us. It would be a serious mistake to assume, going in, that we know in advance how such terms relate.

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, mechanically, by an abstract analogue of a "mere machine";
3. **Rule-following, or execution of an algorithm (RF):** what is involved, and what behavior is thereby produced, in following a set of rules or instructions, such as when making dessert;
4. **Digital state machines (DSM):** the idea of an automata with a finite disjoint set of internally homogeneous machine states—as parodied in the "clunk, clunk, clunk" gait of a 1950's cartoon robot;
5. **Information processing (IP):** what is involved in storing, manipulating, displaying, and otherwise trafficking in information, whatever that might be; and
6. **Physical symbol systems (PSS):** the idea, made famous by Newell and Simon, that, somehow or other, computers interact with (and perhaps also are made of) symbols in a way that depends on their mutual physical embodiment.

No claim is made that this list is exhaustive. In recent years a variety of new construals have started to attract allegiance, including for example:

7. **Interactive agents (IA):** active agents in an embedding environment, that interact with, and communicate with, other agents (and perhaps also with people);
8. **Dynamics (DYN):** the notion of a dynamical system, linear or non-linear, as popularized in discussions of attractors, turbulence, criticality, emergence, etc.;
9. **Complex adaptive systems (CAS):** a notion (primarily associated with the Santa Fe Institute) of self-organising systems that respond to their environment by adjusting their structure so as to survive and prosper.

There are also those who believe that computation's distinctive character is methodological rather than substantive or ontological (thus Agre characterises computer science as an emerging social practice of constructing complex physical implementations of things). In spite of the potential importance of all these alternative suggestions, however, it is the first six, at least to date, that have shouldered the lion's share of responsibility for framing the intellectual debate about the constitutive character of computing.

By far the most important step in getting to the heart of the foundational question, I believe, is to recognize that these 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 long tapes—it is sometimes uncritically thought that all six can be viewed as rough synonyms, as if they were different ways of getting at the same thing. This conflationary tendency is especially rampant in the cognitivist literature, much of which moves around among the lot 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 all six, is false.

Thus the formal symbol manipulation construal (FSM), contrary to widespread assumption, is explicitly characterized in terms of computing's 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 (the not-very-interesting thesis that computation is the manipulation of entities without regard to any issues of interpretation I call **stuff manipulation**). The digital state machine construal (DSM), in contrast, 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 discrete, they would be differentially classified by the two construals. Not only do FSM and DSM *mean* different things, in other words; they have distinct (if overlapping) extensions.

The second and third construal—effective computability (EC) and rule following or algorithm execution (RF)—similarly differ on the crucial issue of semantics. Whereas the effective computability construal, especially in its familiar mathematical guise, seems free of intentional connotation, the idea of rule *following* (as opposed to more ubiquitous rule honouring) seems not only to involve rules or recipes, which presumably do mean something, but also to require some sort of understanding and perhaps commitment on the part of the agent producing the behavior.[7]

Semantics is not the only open issue. It is similarly unclear (as will become apparent below) 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 physical properties, or whether, as most current theoretical discussions suggest, effective computability should be taken as an abstract mathematical notion. The construals also differ on whether they focus on internal structure or external input / output—i.e., on whether (i) they treat computation as fundamentally *a way of being structured or constituted*, so that surface behavior is derivative (the formal symbol manipulation and digital state machine construals are of this type); or whether (ii) the *having of a particular surface behavior* is the essential locus of being computational, with questions about how that is achieved left unspecified and uncared about (effective computability is like this, with rule-following something of an intermediate case).

Not only must the six construals be differentiated one from another, moreover; additional distinctions must be made within each one. Thus the idea of information processing (IP)—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'—needs to be broken down, in turn, into at least four sub-readings, depending on how 'information' is understood:

1. What I will call a **lay** notion, perhaps dating from the nineteenth-century, of something like an abstract, publicly-accessible commodity, carrying a certain degree of impersonal authority;
2. A more current **popular** notion, having to do with the current preoccupation with information technology and the internet;
3. The **quantitative** (and largely semantics-free) notion underlying so-called "information theory," that originated with Shannon & 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

[7]Ref Haugeland.

4. The **semantical** notion of information advocated by Dretske (1981), Barwise & Perry (1983), Halpern (1987), and others, that in contrast to the third deals explicitly with semantic content and veridicality.

By the same token, the formal symbol manipulation (FSM) thesis breaks into positive and negative readings, one having to do with what formality includes (something like syntax or form); the other, with what it excludes (semantics). And so on and so forth.

Clarifying 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. They have certainly underwritten my own work, where I attempt a thorough analysis of each of these six construals.[8]

Yet in a sense these issues are all secondary. For none have the bite of the second question raised at the beginning of this section: of whether any of these enumerated accounts is *right*. Naturally, one has to say just what this means—has to answer the question, that is, "Right of what?," in order to avoid the superficial response: "Of course such and such an analysis is right; that's how computation is *defined!*" This is where the (first) empirical criterion takes hold. "Right of *practice*," is, by my lights, the proper reply. And it is a reply with bite. For when subjected to the empirical demands of real world computing and the conceptual demands of cognitive science, *all six construals fail*—for deep, overlapping, but ultimately distinct, reasons. No one of these six construals, on its own, nor any group, in combination, is strong enough to meet the three criteria laid out above.

3. Prospects

That might seem like a negative conclusion. But it is a consequential one. For the problem is not just that current theories are inadequate. Nor is it just that I have failed, in my thirty-year search, to find a better one. What I have come to believe—and am now prepared to argue—is that I *had* to fail. We will *never* have a theory that meets the three stated criteria. We will never have a theory of computing because *there is nothing there to have a theory of.*

Computers, in the end, turn out to be rather like cars: objects of inesti-

[8] *The Age of Significance: An Essay on the Foundations of Computation and Intentionality, Volumes I–VII*, forthcoming from the MIT Press.

mable social, political, and economic importance, but not destined, per se, to be the subject of deep, intellectually satisfying theory. The problem is that there is nothing sufficiently *special* about computers. In spite of the press, computers in the wild prove, on inspection, not necessarily to be digital, not necessarily to be abstract, not necessarily to be formal—i.e., not to exemplify any characteristic property making them a distinct *subspecies* of the genus "meaningful material system." Computers are dynamic, meaningful, physical artifacts—the best we know how to build. Period. There is no more to say.

As I say, this might seem a dismal conclusion—especially for someone who has spent their professional life looking for a theory of computing. But in fact the opposite is true:

> *That there is no theory of computing (and, hence, no computational theory of mind) is the most optimistic conclusion that anyone involved in the development of computation could possibly have hoped for.*

Seeing it as an optimistic conclusion requires a radical change in perspective. Instead of being a subject matter, warranting its own autonomous theory, computing turns out to be a *site*: an historical occasion on which to see general (unrestricted) issues of meaning and mechanism play out. And it is a site of immense importance—signaling nothing less, I believe, than the end of 300 years of materialist natural science (a study of matter, materials, and mechanism), and the emergence an era of intellectual history of equal importance—an era I label "the age of significance," in which issues of meaning and mattering take an equal place on the intellectual stage with issues matter and mechanism.

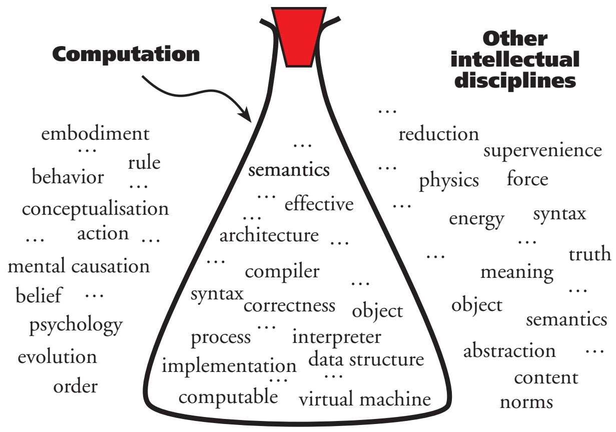

Figure 3a — Computation as Distinct Subject Matter

What computing has given us, in other words, is not a new way of being, but a *laboratory* in which to explore and develop age-old issues—a laboratory of middling complexity, between the frictionless pucks and inclined planes of mechanics, and the rich, full-blooded texture of the human condition, in which to explore the unrestricted interplay of meaning and mechanism.

One way to understand this conclusion is depicted in figure 3. Figure 3a indicates the traditional conception. Computing, depicted as an Erlenmeyer flask, is viewed as a distinct subject matter—taking its place alongside chemistry, say, or history, or physics. It studies a variety of phenomena characterised *as computational*—and thereby thought to be distinct from the rest of scientific or intellectual life. Figure 3b depicts what I believe to be the true nature of the situation. The idea of a theoretically-distinct computational realm has been set aside (the confines of the flask, as it were, peeled back). Issues of classical and current interest in computer science—abstraction, implementation, materiality, modelling, virtual machines, etc.—are allowed to join their natural counterparts (reduction, emergence, supervenience, etc.) from the rest of intellectual inquiry.

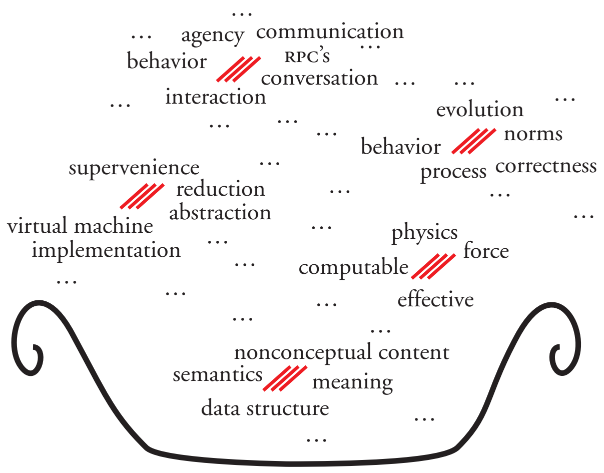

Figure 3b — Computation Unbundled

When I have presented this picture to those involved in theoretical issues in computing, I have been surprised by how readily they seem to agree to it. Perhaps this should not be unexpected. There is, I believe, something relaxing about the reconfiguration. It not only fits the phenomena better than the standard view, but makes more sense of fifty years of experience. But it is no small change. Among other things, it (i) renders vacuous all statements of the form "computers can (or cannot) do α," for arbitrary α; (ii) evacuates the computational theory of mind of intellectual substance; (iii) implies that the theory of computability must either be discarded, or else recognised as being a theory of something else; (iv) implies that no interesting theoretical statement should contain the "c-word"; and (v) challenges the integrity of computer science departments.[9] Nevertheless, I believe it is a change we must embrace. Only when we understand computing as a site, not a subject matter, will we appreciate its true intellectual impact.

Part II — Dialectics

How do these conclusions affect the four dialectics with which we started?

It changes their tenor. I initially identified them as *distinctive characteristics of computing*—as issues, cross-cutting the six (nine, whatever) construals, that gave us a leg up on understanding what computing is. It is evident, however, that all four dialectics (meaning/mechanism, abstract/concrete, static/dynamic, one/many) apply to all sorts of system that are not obviously computational—including, among other things, people. This generality, in turn, makes it natural to suppose that the ways in which the dialectical issues would manifest in computers would (in some way) be *particular to the computational case*.

Thus consider the issue of meaning and mechanism. Many people[10] believe that the computational version of meaning is somehow restricted; that the relationship between intentionality and materiality in the case of computers is in some critical way *different* from the same relation in people, presumably because the *kind of intentionality* that computers exhibit is lesser, or anyway more constrained, than full-blooded human intentionality. For example, such people might think that computers are *formal*, in a way in which people are not. Or that computers intrinsically lack *originality* and *commitment*, in a way that is distinctive of human capacity for significance. It is exactly this kind of assumption that I am concerned to block. To repeat: as far as I have been able to tell, over these 30 years, there is nothing specific to computation—*nothing necessarily true of (all) computers in virtue of their being computers*—that necessitates their being distinctive as regards intentionality, or indeed as being distinctive as regards any other issue. However meaning and mechanism do or can relate, in people or in general, that form of relation will some day or other, I believe, be manifested in at least some computational cases.

[9]Few universities, after all, have departments of cars.

[10]Including some, I would wager, attending this workshop.

What this suggests is that we approach the dialectics differently—or rather, that we approach *computing* differently: as a laboratory in which to study each dialectic as a full-fledged issue in its own right—without limitation or compromise.

Under that rubric, a word about each.[11]

4. Meaning and Mechanism

That computation involves an interplay of meaning and mechanism is perhaps the deepest fact about it. The issue is age-old: how a patch of "physical stuff" can nevertheless represent, mean, wonder, commit, be oriented towards the world.

The dialectic is not one of strict opposition. Pacé Descartes, few think that meaning and mechanism come in two distinct kinds of stuff. Rather—somehow or other, in ways theory must explain—intentional systems in general (and computers in particular) are entities that, while on the one hand "merely physical," nevertheless, presumably in virtue of the *ways* in which they are physical, are at the same time interpreters and dreamers—loci of significance, subject to interpretation. Because of this simultaneous "more than physical" but "no more than physical" character, the dialectic carries something of a tension between the immanent and the transcendent.

I take the meaning side of things—the semantic or intentional character of computing—to be virtually self-evident. It is certainly endemic in the way we talk: *programming languages*, *data* bases, *knowledge representation*, programming language *semantics*, etc. All those terms ('knowledge,' 'language,' 'data,' 'information,' 'variable,' 'term,' 'identifier,' etc.) are intentional—a fact that would be a miraculous coincidence, if computing were not a fundamentally intentional phenomenon. Note too that the computational theory of mind is entirely dependent on the fact that computers, like us, are intentional beings (in most other ways brains and computers are extremely different). Indeed, sans a semantic or interpretive side, it is hard to see what would be distinctive about computing beyond being mere concrete physical stuff.[12]

I take it that the physical or mechanism side of computing is equally ob-

[11]In this pre-conference proceedings I will address only the first two. All four will be addressed in the final paper.

[12]Except perhaps digitality; see the fifth construal, above.

vious. Computers aren't just abstract theoretical posits, or pure ideals. Rather, they have essentially to do with what *works*, what can *happen*, what can be *done*. Agre, as already mentioned, has characterised computer science as essentially no more than an inquiry into unrestricted physical implementation.

How intentional and material existence relate is of course an enormous question, which has preoccupied philosophy for hundreds (if not thousands) of years. On the surface, the two kinds of phenomena are radically distinct. Materiality or mechanism is *causal*, *local*, *energetic* (i.e., physical state change requires energy), and ultimately, because of the nature of underlying physics, both *incremental* and *ahistorical*. Ultimately, our understanding of mechanism must be grounded in something like physical field theory. Phenomena of meaning, interpretation, reference, etc., are stunningly different. Reference, for example, to take just one aspect: leaps across gaps in time, space, and possibility, violating all sorts of physical locality proscriptions; is not energetic at all, except locally (in the sense that no meter, attached to an object, can detect when that object is referred to); is not obviously incremental; traffics in ordinary material ontology, rather than fields (because representation is *representation as*); is normatively governed (involving not only considerations of truth and rationality, but perhaps other norms as well, from lower-level notions of "working vs. broken" to high-level norms of commitment and ethics), and so on and so forth. Indeed, experience with computation should not blind us to the fact that for many centuries, the two phenomena seemed so spectacularly unlike as to defy integration.

Computation, as already suggested, is widely alleged to deal with the meaning / mechanism dialectic in a fashion that is restricted in two critical ways. First, the *kind* of intentionality that computational systems are thought to manifest is often thought to be restricted: to representation or information, for example, as opposed (except in science fiction) to consciousness and full-blooded ethical commitment. Second, these special forms are taken to be further restricted, due to computing's alleged *formality*. Computation's formality, in fact—for example as inscribed in the first (formal symbol manipulation) construal—is taken as key to its ability to resolve what might otherwise be incommensurable. For example, Fodor explicitly characterises (his conception of) the computational theory of mind as a major advance (over an obviously true but rather vapid more general representational theory of mind) exactly because computing is subject to a formality condition.

What does ‘formal’ mean? To answer that in general is deucedly difficult; ‘formal’ is a stubbornly recalcitrant predicate. Much of my 30-year study has focused on it: what ‘formal’ means, what it is to say that computers are formal, whether computer science is or should be a formal discipline, what the nature and applicability is of so-called “formal” methods, etc. But if we restrict ourselves to the first construal, mentioned above—that computers use symbols or process information—we can get some sense of what is at stake.

In particular, the notion of ‘formality’ (in the FSM context) can be broken down into two readings: one positive, one negative. On the positive reading, computing’s active (mechanical) symbol processing is mandated to depend only those symbols’ *form*, or *shape*, or *grammar*—or other effectively potent property. On the negative reading, this same symbol manipulation is militated to proceed *independent of those symbol’s semantics*. This is all supposed to work in the way in which we are taught in logic: one posits a language or set of symbols or expressions with a determinate grammar, over which operations (proof theory or inference) can be defined, without reference to issues of reference, meaning, interpretation.

The problem, in analysing this claim in depth, is to figure out what this double characterisation of formality comes to, without either (i) narrowing it to the extent that it is rendered demonstrably false of real-world computers, or (ii) broadening it to the point of vacuity. The threat of vacuity is heightened by the characterisation’s formulation in terms of “works” or “manipulates.” First, as was suggested above, for the negative claim to have bite, there must be semantics around, for the processing to proceed independently of. Without it, as we saw above, the formal symbol manipulation construal would amount to no more than stuff manipulation. Second, for the positive reading of formality to have substance, the “form” or “shape” or “grammar” must amount to more than the causal or effective or mechanistic properties of the symbols, or else the (positive) claim would amount to no more than a claim that the system works causally (effectively, mechanically) in virtue of its ingredients’ causal (effective, mechanical) properties—which is mere tautology.[13]

The fundamental problem, as I document in my analysis of the FSM con-

[13]So construed, the claim is even weaker than physicalism, since it doesn’t claim that the system is constituted physically, but only that its physical dimension is physical.

strual,[14] is that I have not been able to find any definition of formal that meets these two criteria: i.e., that *restricts* what "formal symbol manipulation" means to something stronger than an obvious consequence of physicalism, but that remains true of real-world computational systems. Problems arise on both sides of the equation: with finding a (true) positive reading that is stronger than "causal and effective," and a negative reading that is true of computation in the wild. As regards the former, it seems easy to find counter-examples to all reasonable suggestions restricting "form" to anything less than "any possible causally efficacious property." On the latter front, the problem is that the only sorts of systems for which a negative reading of formality seems ultimately defensible are systems that are *disconnected* from their subject matter, in the way that is always imagined for theorem provers, logical axiomatisations, etc. Real-world computers, however, are typically *engaged* in their subject matters: they participate, causally and effectively, in the very domains that their symbols represent. It is that real-world involvement, ultimately, that defeats the negative reading.

So formality fails, positive and negative. That fits with the overarching claim made earlier: that computation is not a *special* kind of symbol manipulation; it is symbol manipulation *tout court*. But—and this is where the recognition of computing as a "site" has impact—that failure of formality doesn't mean that we have nothing to learn from our computational experience about the relationship between meaning and mechanism. On the contrary: we can view the failure of formality (and the general deconstruction of computing it exemplifies) as a boon: by discarding it, we rid ourselves of a theoretical obstacle that has blocked us from understanding the power and generality of that very experience.

To see how this goes, we need to reconstruct the work that was being done, in the computational setting, by the alleged commitment to formality. To describe that here requires compressing a book into two paragraphs,[15] but I hope I can at least convey a flavour of the argument.

The *positive* reading of formality (that systems operate in terms of syntax, form, shape) I reconstruct as a commitment to the causal efficacy of how systems work—i.e., as an endorsement of a critical aspect of physicalism. Except

[14] *The Age of Significance: Volume II • Formal Symbol Manipulation*

[15] «Ref AOS·II.»

that it is not level-neutral; it is not merely the claim that the system can be understood as a coherent mechanism at *some* (perhaps a very low) level, but that it can be so understood—i.e., that it can be coherently analysed qua a causal mechanism—*at the same level as that at which it is semantically interpreted*. So construed, the claim can be broken into two parts. The first part, which merely affirms the importance of materiality and effective mechanism, has historically been substantial because formal systems have traditionally been analysed in theoretical contexts that take them to be abstract. That is: formality (in its positive reading) can be seen as *the shadow of materiality* in a context that has tried to eschew embodiment (more on this under the second dialectic). The second part of the claim, that the system can be coherently understood as mechanical at the same level at which it is semantically evaluable, is more contentious. Recent research on emergence and on semantic externalism (somewhat unlikely bedfellows) challenges this claim, though that challenge is in turn partly countered by the extraordinary flexibility that computer science has achieved in implementing all kinds of abstract machines on top of each other. My own view is that it is too early to make a call on how this issue will resolve. Note, that, that by ridding ourselves of the distraction of formality, we can see this "causal efficacy and semantic interpretation at the same level" as something of a denial of Davidson's anomalous monism.

The negative reading of formality claims that computational systems operate *independent* of semantics. There are a host of different things that can mean, but in the end, because of the (afore-mentioned involvement in their subject matters), all plausible readings prove too strong; they are false of computation in the wild—primarily because of such systems' participatory engagement in their subject matters. But once again, that does not mean there are no general lessons to be learned. On the contrary, one positive lesson has to do with the general *inefficacy of the semantic:* a recognition that semantical properties (such as reference and truth, paradigmatically, but even meaning, at least in a "wide" sense) are not effective. Systems cannot always, and perhaps not ever, do what they do in virtue of the semantical properties manifested by their internal parts.

This may seem like restatement of the initial claim: if the system does not work in virtue of the semantical properties, does that not imply that it does work independently of them? The answer depends on the strength of notion of independence. Some will argue that "does not work in virtue of" entails "works independently of", at least on a logical reading of independence. But I believe that is misleading. There is a world of difference between "not dependent" and "independent"—a vast intermediate terrain of *partial* dependence, or, to adopt a more useful characterisation, *partial interdependence*. What the thick engagement[16] of real-world systems with their semantic domains shows us, I believe, is how the two prongs of the dilemma—the limitations and constraints of effective material operation, which constrain the system to operate, mechanically, in terms of its internal machinery and the physical state of the impinging environment, on the one hand, and the locality-transcending reach of semantics and reference, on the other—lead a fully engaged system to substantial and authentic intentional engagement with that domain. And to repeat: these are lessons that an unwarranted commitment to formality and computing would have blinded us to.[17]

There are a thousand other things to say; this is the tip of a very large iceberg. But we are running out of time—and I have touched on only the first of the four dialectics. What I hope is that over this workshop we will have time to explore in more details some of these remarkable consequences of the concrete technical practice that lies in front of us, waiting to be disclosed.

5. Abstract and concrete

Given 300 years of scientific success in explaining the properties of physical phenomena, one might expect that the "mechanism" side of the primary dialectic would be in good shape, and that all the theoretical difficulties would have to do with meaning. But it is not so. Explaining the mechanism side of the primary dialectic has proved surprisingly recalcitrant. In fact virtually all of 20th century computer science has been devoted to this topic—at the expense of the semantical. Where meaning has been studied, when it has been studied at all, is in logic, cognitive science, and the philosophy of mind. (It may seem odd to say that computer science has not studied semantics, given the prominence of what is called "programming language semantics"—but that is a terminological distraction, I believe. See the sidebar on "Program vs.

[16]Compare: Haugeland's intimate engagement «ref».

[17]Fortunately, industry is not shackled by theoretical ideals. Commercial software, I believe, gives us example after example of thick in-the-world intentional participation.

Process Semantics".)

The basic question is how we are to understand mechanism: what the powers and limitations of mechanisms are; what can be done, with what resources, in what amounts of time; and what cannot be done at all. On the face of it, one might expect that the answer to such questions would depend on the particular materials out of which one makes things. Famously, however, in what is taken to be one of the century's great theoretical advances, early work in recursion and computability theory lifted the notion of mechanism away from concrete physical considerations and (allegedly) formulated it abstractly (for example, in the notion of a Turing machine). The basic motivation for this abstracting stance is straightforward: the "very same computation," it is argued, can be implemented on arbitrary different machines, made of silicon, vacuum tubes, DNA—even Rube Goldberg arrangements of Tinker Toy.[18] What matters about the computation thus implemented, it is argued, must therefore be at a more abstract level, above such mundane material considerations.[19]

Recently, however, there have been suggestions from many different quarters of computing suggesting that this abstract perspective may have gone too far. Circuit designers have come to realise that issues of three-dimensional geometry may ultimately have more to do with the limits on what can and what cannot be done than pure abstract topology. At the deepest levels, computer scientists are collaborating with quantum physicists to define a purely physical notion of information, in an effort to fuse the foundations of the

[18]In cognitive science and philosophy of mind this is known as the *multiple realisability argument*.

[19]Of various reasons why logic and theoretical computer science treat computability results abstractly, the most famous have to do with multiple realisability. But there are other reasons, some of which are surely at least as important. One big one is that logic and computability theory developed, historically, out of concerns with metamathematics, making an abstract perspective more natural. Curiously enough, this remains true even when the focus is on language, not just on the (mathematical) entities that the language is about. The problem is that the languages used to represent mathematical subject matters are by and large context-independent, which turns out to entail that one can frame results about such languages purely in terms of types, without regard to concrete specific facts about individual tokens (the way one needs to do when treating indexical expressions, for example). It has thus proved convenient, I believe, to deal with states, marks, etc., as *abstract individuals*, whereas in my own view they are more properly understood as *concrete types* (i.e., types of concrete things).

two fields.[20] And in AI and cognitive science, the abstract and rationalist models of computing inscribed in the classical GOFAI model of cognition are being roundly rejected, in favour of embodied, dynamical alternatives, that take the concrete physical existence of the cognitive agent much more seriously. In sum, we are in the midst of something of a **re-materialisation** of computing, aiming to undo the abstracting excesses of the logical tradition from which theoretical computer science inherited most of its intellectual tools.

Commensurate with this general shift, the primary result of my analysis of the second effective computability (EC) construal of computing is a claim that the fundamental computability limits—the constitutive constraints on which theoretical computer science rests—in spite of tradition, derive from the physical. That is: I claim that both absolute computability limits, as in Gödel's incompleteness results, the unsolvability of the halting problem, etc., and relative computability limits, as in complexity theory, the difficulty of deciding classes of formulae, etc., derive directly from concrete, material constraints on underlying mechanisms. It is not just that the results are *framed* mathematically; so are (at least many) results in physics and chemistry. Rather, the claim is that, as in physics and chemistry, the mathematical structures in terms of which we frame them are not the real subject matter, but models of the subject matter—which is to say, models of physical phenomena. Present theoretic practice notwithstanding, the subject matter of theoretical computer science is by my lights entirely concrete.

Needless to say, this is a contentious claim. Informally, in my experience, it divides people by discipline: I have yet to meet a logician who believes it, or working programmer who disbelieves it. What makes the issues subtle (and interesting) is the fact that computability results are (i) not specific to any *particular* material substrate (factoring primes is approximately equally difficult whether one uses vacuum tubes, silicon transistors, or tinker toys), and (ii) not expressed in physical *units* (kilograms, ergs, etc.), suggesting that they must actually be about numeric quantities. But that suggestion is illusory. It turns out that if one changes the physics of the realizing substrate, one can change complexity results at will. Intimations of this were recognised as early

[20] «Refer to the identification of the loss of a bit of information with a measurable quantity of heat.»

as in the 1930s by Gandy, who showed that the absolute computability results depended in immediate and subtle ways on the character of the physical mechanisms on which they were assumed to be implemented.

Some of the arguments I advance in support of this concrete reading of computability are negative: that *not* recognising and understanding the physical nature of effectiveness yields several unhappy results: one can solve the halting problem, cannot explain the critical (but never explained) notion of a "reasonable encoding," etc. The lion's share of the argument, however, rests on positive results: that if one *does* recognise the concrete nature of effectiveness, one can (among other things):

1. Explain the notion of a reasonable encoding—including (i) what the constraints on reasonable encodings are, (ii) why encodings play such a critical role in computability theory, and even (iii) why the notion of a reasonable encoding has received so little theoretical attention;
2. Make sense of the rise of Girard's linear logic, and computer science's interest in intuitionistic type theory and constructive mathematics;
3. Predict the proposed fusion of foundational theories of quantum mechanics and computer science-based theories of information;
4. Make sense of why physicists (not just computer scientists) are interested in super-Turing computability, continuous models of computation, quantum computing, etc.; and
5. Resolve otherwise unexplicated tensions between what is *real* and what is *virtual* (e.g., in popular conceptions of computational technology).

I would be the first to admit, though, that reformulating our understanding of (what is known as) computability in concrete, material terms, in the way I recommend, is an enormous intellectual task. My guess is that it will take at least 50 years for the transformation to take place. For example, all familiar computability results—that one cannot in general decide whether a Turing machine will halt on an arbitrary input, that factoring primes is hard, etc.—will require complete conceptual reformulation, as issues about *mechanisms*, not issues about *numbers* or *decisions*. The (so-called) theory of effective computability that I have indicted, several times, as a theory of computing, will have to be reconstructed as a theory of *effectiveness* (and not a theory of *computing* at all—as suggested at the end of Part I), because it deals with only the first (mechanism) arm of the primary dialectic, not with the second (meaning, semantics, reference, truth, etc.). What it is, I argue—something that history will eventually recognise it as—is neither more nor less than a **mathematical theory of causality**: that is, a theory of what can be done, in what time and with what resources, by what sorts of arrangements of concrete, physical stuff. That such a theory should be framed at *some* level of abstractness, away from very specific concerns having to do with particular materials, is entirely to be expected. It is for this reason that I have dubbed the properties that the theory traffics in *effective* properties, rather than physical properties; they are properties that systems (or states) can *do consequential work in virtue of possessing*.[21]

One final remark. A proponent of embodied cognition might argue that even if we do reconstruct computability theory as a theory of causality, it will still be too abstract: that in order to understand in-the-wild cognition, one needs to understand not only relatively abstract causal properties of the system, but quite concrete properties (such as heft and materials)—e.g., in order to understand rhythm and dynamic movement. That may be, but as already indicated, there is every indication in theoretical computer science that theory is rapidly being refined so as to deal with more and more direct physical parameters (in order, among other reasons, to treat issues of three-dimensional packaging and real-time computing). Moreover, somewhat counter to this point, the embodied cognition movement has to be interested in bodies and materials at *some* level of abstraction. Suppose one were to replace the control circuit for the muscles of an animal with an electronic souped-up version; what matters, presumably, even to the most materially-oriented theorist, is that the signals match, that power be supplied, that the right function be computed in real-time, etc. There are questions of whether such implants could work—and how much of our cognitive facilities could be upgraded in this way. But virtually no one thinks that a brain implant

[21]It is also unclear exactly what it is to be a physical property. Being a million light-years from Alpha Centauri is presumably a physical property, but not an effective one; it would be impossible, at least in any remotely practical sense, to build a device that could "detect" the exemplification of this property.

would literally have to be made of DNA-based neurons, in order to function in a "materially" appropriate way.

Put it this way: neurophysiology and the theory of effective computability are climbing up the same mountain, even if from different sides.

6. Static and Dynamic

7. The Many and the One

8. Relations

These last three sections, which were not completed in time for the proceedings, will be addressed in the conference presentation and included in the final paper. Sections 6 and 7 deal with the third and fourth dialectic, respectively. Section 8 will address a variety of ways in which the four dialectics are intertwined (no claim is being made that they are independent).

8. Conclusion

I do not claim that these four dialectics *constitute* computing—or that they constitute the realm of meaning and mechanism, subject matter of the age of significance. That is, I am not claiming that an adequate account of meaningful (intentional) concrete systems would be a four-dimensional theory framed along these four axes. Rather, the point is more modest: that these four issues permeate computing; that any satisfactory theory must address them; that they provide us with a useful framework in terms of which to understand how different theories compare.

Needless to say, other issues would have to be addressed as well, in order to do justice to the full range of real-world computational phenomena, including for example the relation between a system described at some level of abstraction (i.e., a system under a description) and that same system described at some other level of abstraction—an issue that implicates such critical notions as reduction, supervenience, abstraction, emergence, and implementation. But those are issues for another day.

The fact that there are always more issues to address is perhaps the most durable part of studying real-world computing. If one operates under the misconception that computing is a subject matter, that seemingly endless proliferation of concerns can be discouraging; one despairs of ever being able to "rein the subject matter in." One of the great benefits of recognising that computing is an unbounded site, where we are experimenting with building arbitrarily complex intentional systems, is that one *expects* that new issues will forever emerge. Far from being discouraging, this experience—once one approaches the phenomenon from the right perspective—can be seen as testament to computing's unending fertility.

——— *end of file* ——————