From E&M to M&E[1]

A journey through the landscape of computing

Brian Cantwell Smith[2]
University of Toronto

Not only are the philosophy of computing and the philosophy of information new fields, still theoretically unstable, but the subject matters they span are exceptionally broad. "Information" covers so many phenomena as to be threatened by vacuity—though that hasn't deterred people from using it as an explanatory concept in fields as diverse as biology, computer science, medicine, journalism, electrical engineering, literature, the arts. Computation is narrower, and seems better understood, in part because of half a century's work on mathematical theories of computability. But here I believe appearances are misleading. Not only do we not understand computing as well as is generally thought, I will argue, but making progress will require upending all sorts of fundamental assumptions in ontology, epistemology, and even metaphysics.

This combination of newness and breadth means that no contributor to this volume can assure the reader that the path they have traveled through the landscape may not be due as much to their own philosophical predilections as to any intrinsic geography. So there is merit to Floridi's suggestion that we start with biographic details. However it also means that all writing in these areas (my own included) is liable to fall prey to Isaiah Berlin's challenge that "writing is amateur when you learn about the author, not about the subject matter."

Forewarned is forearmed.

I • Origins

My own interest stems from my first semester at university, when an IBM 360/44 was delivered into the basement of the Oberlin College physics department. Riven by a naïve version of C P Snow's two-culture dilemma, I wrestled with whether to drop physics and major in religion, debated politics with anyone

[1]"E&M" is physics' moniker for "electricity and magnetism," the field from which I entered computing. "M&E" is philosophy's parallel epithet for "metaphysics and epistemology," the landscape to which my travels through computation have led.

[2]To be published in Luciano Floridi, ed., *The Philosophy of Computing and Information: Five Questions*. Version of Jan 31, 2008. Please send comments to: brian.cantwell.smith@utoronto.ca

who was awake, and spent the remainder of my nights ferrying stuffed boxes of punched cards back and forth to the operator's window at the Computing Center. Crazed, yes; but it made a kind of manic sense. Knowing nothing of hermetic methods or intellectual precursor, I was possessed by a conviction that the power and elegance of science, the gravity and richness of politics and religion, and the intensity of intimate human communion were ultimately more similar than they were different.

Within two months I had made two life-altering decisions. First, I vowed to dig deep enough to get to the place where these superficially different perspectives could be understood, if not as "one," then at least as *integral*—as part of a single encompassing reality. Second, at a more pedestrian level, I asked my physics professor for 6 weeks off from doing problem sets, to figure out whether the school's new computer might help with this quest. What I wanted to know, I told him, was whether computing could be understood with all the power and insight and elegance that I loved in the sciences, but nevertheless do justice, in a way no prior scientific account ever had, to the richness and complexity of the human condition.

It was a classic sophomoric venture: wisdom shot through with foolishness. All told, it wasn't a bad question. But six weeks turned into forty years.

The first results stemmed from those long nights of debugging. Inchoately at first, but more articulately as the years went by, I came to believe that the understanding of computing I was deriving from concrete engagement—not just at Oberlin, but later writing operating systems, implementing data bases, designing programming languages—was not accounted for by what was being *taught* about computing in the nascent field of computer science. The problem wasn't just that theories idealized, or ignored practical realities one had to come to grips with in real-world settings. As much is true of any engineering practice. Rather, I could never shake the feeling that the accounts were profoundly wrong, misguided at their core—"missing" what mattered most about the territory we were tacitly and somewhat blindly exploring.

In parallel, motivated by an interest in people and mind, I was drawn into artificial intelligence (AI) and cognitive science, initiatives whose fortunes were on the rise, as society grappled with the monumental idea that computing was not just a technology of disruptive impact, but also a powerful *idea*—perhaps even one that applied to us. *Maybe we, too, were computers.* Debates raged, with endorsements rung from the MIT, Carnegie Mellon, and Stanford AI laboratories,[3] critiques lobbed back by Weizenbaum, Dreyfus, and Searle,[4] and more speculative analyses taken up across the philosophy of mind.[5]

Naturally, I wanted to formulate my own position. But I was blocked by my underlying sense of discrepancy between *how we thought computers worked* and my blood-and-bones intuitions as to *how they actually worked.* The situation is depicted in figure 1. Debate on what came to be known as *the computational theory of mind* (CTOM) was presumed to have the structure labeled α. What it is to be a *computer* was assumed to be uncontentiously formulated in the "received" theory of computation, labeled θ_c in the diagram. At stake was how to understand the *mind*: θ_m. The substance of the CTOM was taken to be the thesis that $\theta_m \approx \theta_c$.[6]

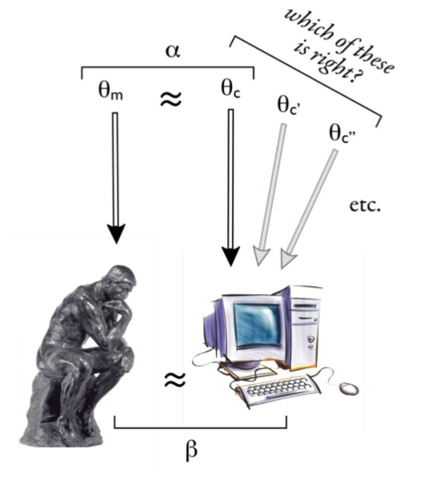

Figure 1—Computational Theory of Mind

My problem was straightforward. Fundamentally, I took the CTOM not to be a theory-laden proposition, in the sense of framing or resting on a specific hypothesis θ_c about what comput-

[3]Particularly Marvin Minsky & Seymour Papert at MIT, Allen Newell and Herbert Simon at Carnegie Mellon (CMU), and John McCarthy at Stanford.

[4]Especially Joseph Weizenbaum's ELIZA program [1966], Hubert Dreyfus' *What Computers Can't Do: A Critique of Artificial Intelligence* [1972], and John Searle's "Chinese Room" thought experiment [1980].

[5]E.g., Haugeland's *Mind Design* [1981], and *Artificial Intelligence: The Very Idea* [1985]. Mid 19th-century philosophical discussions of computing were primarily conducted in the philosophy of mathematics, pursuant to Gödel's proof and Turing's computability results; by the end of the century, the debate had moved to the philosophy of mind. It is only now that an authentic philosophy of computing is coming into its own.

[6]Not, of course, that anyone thought that *all* computers were minds ($M = C$). Even if all minds are computational, the class of computers is larger, and so clearly $M \subset C$. This raises the question of how what was *specific* to mind, and how that would be articulated. The complexity of the questions was rarely explicitly addressed, but it was presumed that the restriction of C to M would also be expressed *in computational terms*—as opposed, say, to minds being those computers that "weigh more than 1 lb but less than 10," a restriction that leaves minds as computers, but where the restriction itself is not, as it were, a "computational" restriction, not being framed in terms of a property (weight) that is itself a "computational property."

ers are, independent of whether θ_c held of real-world computers. Rather, I took it to have an ostensive or "transparent" character (β): 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 or might be computers.[7] It wouldn't be interesting, I felt (this was no attempt to vindicate Weizenbaum, Dreyfus or Searle), if it emerged that, sure enough, θ_c wasn't true of people, but θ_c *wasn't true of the IBM 360, ABS brake systems, or my word processor, either*. Suppose, in particular, as I suspected, that θ_c was not the right theory of computing, but instead that $\theta_{c'}$, or $\theta_{c''}$, or some other account, were "correct" or anyway to be preferred?[8] Then the only interesting question, I believed, was whether $\theta_{c'}$ or $\theta_{c''}$ (or whatever) held of people.

A myriad challenges can be raised against this approach, including: that an "empirical" stance cannot be right, because θ_c (i.e., the accepted mathematical theory of computation and computability) is *how computing is defined*; that because we build computers, we *must* understand them; etc. While I disagree with all of this, this is not the place to address it. The point is simply that I believed (i) that the only interesting version of the CTOM needed a theory that did justice to computing, and (ii) that θ_c was not it. And so, around the mid 1970s, I took up the project that occupied me for the next 25 years: to *figure out what computing is* (i.e., which variant of θ_c is right), at a level of depth strong enough to found an adequate theory of computing, and richly enough articulated to support substantive debate about a relevant computational theory of mind.

Before I could address the question of whether the CTOM was true, that is, I needed to know what it said.

By 1972 I had moved to MIT, an epicenter of AI and cognitive science. Instead of entering those programs directly, I first enrolled as a "social inquiry" major—reflecting my interest in assessing, rather than embracing, the CTOM. But the same problem of inadequacy in reigning conceptions of computing impeded my participation in that fledgling STS program—for example, the assumption that computing is a *technology*, as opposed, say, to a form of art or sculpture. Recognizing that insight could only come from substantial engagement, I transferred to the Artificial Intelligence Laboratory for the remainder of my education.

[7]Cf. footnote 4.

[8]Whatever "correct" comes to, whether that is even the right term, etc. For simplicity, I have phrased the issue conservatively.

2 • Preliminaries

Philosophy of computing is in its infancy. Whether history will even notice us I do not know, but we have certainly just scratched its surface. Take a dozen terms of the computational art: *program, process, algorithm, symbol, data structure, implementation, architecture, complexity, object-oriented, user-friendly, nondeterminism,* and *procedural.* Every one remains unreconstructed—some more so than others, but all to an extent that five minutes in an undergraduate class is enough to raise questions that outstrip contemporary comprehension. Even such fundamental notions as *being computational, carrying information, being algorithmic or effective,* etc., remain open—no one knows whether they are: *intrinsic,* like mass and momentum; *abstract,* like numbers or types; *relational,* like being well-loved; or require *external ascription* or *interpretation,* like the meanings of books and text.[9]

I do not say this to be negative. On the contrary, the inchoate state of our understanding greatly energizes the field. It is like participating in the early days of physics. Graduate students can still read everything that has been written, and set out to explore largely uncharted intellectual realms. Critical issues are at stake—not just fundamental ones of meaning, mechanism, and reality—but also such notions as credibility, authenticity, engagement, and the like.

Nevertheless, the modest state of the art does suggest that students enter the field with some humility, lest they be misled into taking more for granted than is warranted. Four cautions strike me as especially important:

C1 • Computers, computing, computation: It is essential not to assume pre-theoretically any particular conception of—or distinction among—such familiar notions as *computer, computing, computation, computable,* etc. One such view has become something of a commonplace in computer science: that *computations,* viewed as abstract objects, are the entities of theoretical interest; and that *computers,* merely physical devices that realize or implement computations, are of no theoretical significance (no matter how economically and pragmatically consequential). This is the stance immortalized in Dijkstra's famous claim that "computer science is no more about computers than astronomy is about telescopes." But any substance to that blunt pronouncement,[10] including the distinctions it is

[9]Everyone, including I, can raise objections to every one of these examples, and to the four-way typology. That is the point. The intellectual structure of the inquiry is still up for grabs.
[10]With which, as it happens but perhaps not surprisingly, I disagree.

framed in terms of, depend on a theoretical framework the adequacy of which should be in question in any foundational analysis.[11]

C2 · Equivalence: It is critical not to give undue weight, especially conceptual weight, to the famous equivalence proofs underlying mathematical computability—proofs according to which various different "models" of computing (Turing machines, the λ-calculus, Kleene's recursion equations, etc.[12]) are shown to "compute" the same class of mathematical functions. Not only is the legitimacy of these proofs rarely questioned; it is also common to assume—falsely, in my view—that they show that the different models are "equivalent" for other purposes as well. Some illustrative problems:

a. As in C1, the proofs rely on a conception of what it is to "compute"—a notion that should be questioned, not assumed, in a foundational account. To assume such "post-theoretic" equivalences in advance will inevitably prejudice, and at worst render circular, any account of computing based on them.

b. The notion of "compute" on which the equivalence proofs rely is extremely restricted. Issues of input and output, and any other form of interaction or engagement with the world, are not so much ignored as banished—kept outside the framework. No mention is made of how the tape is initialized, how the results are "read out" (or interpreted; see C4), or anything of the sort. Time and timing are similarly dismissed. While complexity analyses pay some attention to resources, the claim that a universal machine can do "anything"—at least, "anything that can be done by machine"—is excruciatingly narrow. Tap out the differences among rhumba, reggae, and bebop? Make a cup of coffee? "Out of bounds!" is the standard reply. But who said so—and why? These are things a philosophical analysis should explain, not presume.

c. Conversely, the equivalence metric used in the equivalence proofs is extraordinarily broad—so broad as to sweep under the rug virtually every distinction that might be relevant to a theory of mind: how

[11]Note the contrast with cognitive science and philosophy of mind, which (especially in the analytic tradition) used to view "mind" as essentially independent of body—a dualism that has come crashing down in recent years.

[12]Paradigmatically, devices of minimal structure given access to indefinite storage.

the device works; whether the resulting computation is intrinsic, ascribed, relative, relational, etc.; how long it would take to run; and so on. The standard way one shows that one machine can "do the same thing" as another, in fact, is to have the first machine *model* or *simulate* the second—the very distinction on which Searle based his critical distinction between "weak" and "strong" AI.[13] Distinctions on which competing theories of mind are distinguished—behaviourism, representationalism, type or token reductionism, materialism, etc.—are similarly obliterated in the quest for isomorphism.

d. All semantic issues, about meaning and interpretation, are again ignored or banned. In his original work on information theory Shannon was particularly articulate about this setting aside of issues of meaning and content; for reasons described below, the situation in computing is more complex. But independent of the use of *words*, fundamental issues of how systems signify, represent, carry information about, are interpreted as, or otherwise relate to the world around them are not addressed by any received theory of computing.

C3 • Semantic Soup: In days of Ptolemaic and pre-Copernican astronomy, it was easy to distinguish among the various accoutrements of inquiry: *theory, experiment, equipment, model, representation, subject matter,* etc. Theories were viewed as abstract; representations were written down, probably on parchment; models, such as brass orreries, likely sat on tables; celestial subject matters were a long way away. In computational times, however, one encounters claims that instances of all these categories are *of the same kind*: computational processes of one sort or other.[14] Even in Turing's original paper, distinctions among numbers, representations of numbers, and numeric models are conflated after just a few pages. The mathematical proofs mentioned above, along with such kin as Gödel's incompleteness theorems, category theory, and the like, identify (i.e., conflate) all manner of isomorphic things. Current writers sometimes muse about the overlap,[15] but by and large it receives little at-

[13]Just because ***x*** simulates ***y***, that doesn't mean that ***x** is **y***.

[14]I am not saying a theory *can* be a computation (as opposed to something more abstract); merely, that some people claim so.

[15]Cf. Edelman's ironic comment that he had validated his emphatically *non-computational* model by "implementing it on a computer." «ref»

tention. The caution, here, is not so much an injunction *not to do* this or that, but to keep a strict eye on the soup of semantic relationships in which computational systems simmer, lest the intentional character of the phenomenon dissolve from view.

C4 • Mathematics: In part because of the prior three cautions, I enjoin students never to use mathematical examples as paradigmatic illustrations of computing, or as case studies on top of which to develop a general account. Numbers, numerals, mathematical models, and the like are simply too easy to confuse or conflate for it to be possible to "extract" the true nature of what is going on. Not only that; people's philosophies of mathematics differ by more than the issues at stake in philosophy of computing and/or philosophy of mind. Some people take numbers to be concepts; others, to be Platonic abstractions; still others, to be numerals or expressions; etc. How can one forge a cogent philosophy of computing in the face of such ontological profusion? Better to pervert Gertrude Stein to our purposes: "Forget numbers; think about potatoes."

By way of preparation, especially for those new to the field, two additional observations need to be added to these four cautions:[16]

P1 • Terminological Archeology: Much of the theoretical vocabulary we use to study computing was not invented *de novo*. A great many terms of art were borrowed from logic and metamathematics—the areas in which Turing, Kleene, and other computational progenitors worked. Thus such notions as *syntax*, *semantics*, *symbol*, *identifier*, *variable*, *reference*, *interpretation*, *model*, etc., were used technically in logic long before they were pressed into computational service. This overlap has generated more confusion, I believe, than has been adequately recognized.

Searle's two arguments against the possibility of artificial intelligence are striking examples: (i) his "Chinese Room" argument, that semantics does not inhere in syntax; and (ii) his parallel argument that syntax does not inhere in physics.[17] Searle was trained as a philosopher, and would have learned the words 'syntax,' 'semantics,' 'formal,' etc., from logic. To a person, as far as I know, computer scientists, on reading his arguments, feel that Searle "just doesn't get it." What I have told stu-

[16]It will be obvious later why these two deserve mention here.
[17]Chapter 9 of *Rediscovering the Mind*, MIT Press, 1992.

dents for more than 20 years, however, is that *Searle would have been right, if the words meant what he was taught that they mean*—if, that is, by 'syntax' and 'semantics,' computer scientists meant what the people they took those words from (i.e., logicians) meant by those terms. This is not to excuse Searle, whose conclusions I am not endorsing;[18] but it does throw down a gauntlet that we say, *in language that non-computer scientists can understand*, what computing is. Yet another reason why the philosophy of computing is so important.

P2 · Interdisciplinary Theory: Finally, it pays to attend to the relations between substantive issues that arise in computing and allied questions addressed in other fields—especially as the place of computing in the overall intellectual landscape is not yet well understood or agreed. Just one example: computer science has extensive vocabulary to talk about the relation between one system understood as "an α" and that very same system understood as "a β"—i.e., as we say, *one and the same system analysed at different levels of abstraction*. As well as using such basic terms as 'implementation' and 'realization,' computational discussions involve such notions as *abstraction, modularity, "black-box" and "grey-box" implementation boundaries, importation, exportation, interoperability protocols, interfaces* (including APIs), etc. Philosophy of mind and philosophy of science have developed their own theoretical apparatus to deal with what looks at first blush to be the same subject—under such terms as *type-* and *token-reduction,* (local and global) *supervenience, multiple realisability,* etc.

For years I have offered to supervise a doctoral student to conduct a theoretical analysis of trans-disciplinary vocabulary in this or various allied areas, since I am not aware of any other systematic investigation of how the two analytic frameworks relate. No takers so far, but the offer remains open.

Enough preliminaries. Once the land is cleared, the project of developing an adequate philosophy of computing opens up into something like Frege's investigation of number—except that the empirical commitment requires maintaining focus on concrete, in-the-world phenomena. In that way it is also reminiscent of

[18]I agree with him, as it happens, both that syntax does not inhere in physics, and that semantics does not inhere in syntax—at least on local interpretations of all those words. Where I disagree with him is on the underlying assumption that computation is syntactic.

questions in the foundations of physics: about meaning, interpretation, measurement, and reality.

It does rather mean starting from scratch. But such is the nature of the enterprise.

3 • Project

Given these considerations, how can one proceed? My approach has been to intersect three cross-cutting "axes" along which computation has historically been analysed—generating something like an informal coordinate system in terms of which to map the computational territory. In no way do I endorse the resulting cartography as theoretically sound, or even as particularly coherent. By the time I am done, in fact, I discard every one of these distinctions, or reconfigure them beyond recognition. Still, the system pays its way as an initial guide.

3a — Construals

The first axis enumerates seven "construals" of computing, as I put it, that have variously held sway in our intellectual discourse:

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 independent of 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 behaviour is thereby produced, in following a set of rules or instructions, such as when making dessert;
4. *Calculation of a Function* (FUN): the behaviour, 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 1950's cartoon robot;
6. *Information Processing* (IP): what is involved in storing, manipulating, displaying, and otherwise trafficking in information, whatever information might be;

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.

I don't claim this list is exhaustive. Several more have recently made it onto the scene: non-linear dynamics, complex adaptive systems, a view of computing in terms of interacting agents, and so forth—all of which could be used to extend the list. Contrapuntally, a host of familiar ideas must be set aside as inappropriate for foundational duty: (i) demeaning characterisations, that computing is *just* something or other (machine, mechanism, artefact, attributed, etc.); (ii) negative construals, such as that computing is *not* some way (conscious, original, alive, and so on); and (iii) "higher-order" or adverbial specifications, such as *abstract*, *universal*, *formal*, etc., which only gain traction against some presumed prior property. The members of all three categories implicitly rely on another conception of computing, in order to have any substance.[19] But leaving such complexifications aside, it is the seven listed above—what I call the **classic construals**—that, at least to date, have shouldered the weight of the intellectual debate.

It is critical to recognise that all seven construals are both intensionally (conceptually) and extensionally distinct. In part because of their great familiarity, and in part because "real" computers apparently exemplify more than one of them, but perhaps especially because of the pernicious influence of those pesky equivalence proofs, it is often thought that the seven are roughly synonymous. This conflationary tendency has been especially rampant in cognitive science and philosophy of mind, both of which tend to move around among the seven with abandon. But to do so is a mistake. The supposition that any two of these construals amount to the same thing, let alone the whole group, is simply false.[20]

[19]How do we know that computers are just machines, not conscious, etc.? Only if we have some other account of what they are like, from which such a conclusion could then be derived.

[20]Formal symbol manipulation is explicitly characterized in terms of a semantic aspect of computation, for example, if for no other reason than that without it there would be no warrant in calling it *symbol* manipulation—to say nothing of there being nothing for it to work independently of. The digital state machine construal, in contrast, makes no such reference to semantic properties. If a Lincoln-log contraption were digital but not symbolic, and a continuous symbol machine 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 have overlapping but distinct extensions.

The effective computability and algorithm execution construals also differ on semantics.

Clarifying the issues raised in these construals, bringing salient assumptions to the fore, showing where they agree and where they differ, tracing the roles they have played in computing's first century—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 reason we are interested in the set in the first place: whether any of the enumerated accounts is *right*.

That question, too, must be addressed: to what jury a proposed theory of computing should be held accountable. But for now let me cut straight to the chase: *not one is correct*. Forty years after that freshman year in college, I am prepared to argue that, when subjected to the empirical demands of practice and the conceptual demands of theory, *all seven construals fail*—for deep, overlapping, but distinct, reasons. No one of them, nor any group in combination, is adequate to meet the requirements of a foundational account.

3b — Dialectics

To understand the reason for this failure, and grasp the picture of computing that comes out of it, it helps to identify the other two "axes" I use as an initial guide to the territory—both of which cross-cut the first division into construals.

Whereas effective computability seems free of intentional connotation, the idea of algorithm execution seems not only to involve rules or recipes, which presumably do mean something, but also to require something like "understanding" or at least "semantic compliance" on the part of the agent producing the behavior. It is also unclear whether the notions of "machine" and "effectiveness" refer to causal powers, material realization, or other physical properties—or, as current theoretical discussions suggest, effective computability should be taken as an abstract mathematical notion. (This is no small question; if we do not yet understand the *mind/body problem for machines*, how can we expect computational metaphors to help us in the case of people?) The construals also differ on whether they focus on internal structure or on input/output—i.e., on whether (i) they treat computation as *a way of being structured or constituted*, so that surface behavior is derivative (FSM and DSM), or whether the *having of a particular surface behavior* is the essential locus of computationality, with questions about how that is achieved left unspecified and uncared about (EC, perhaps ALG).

Not only must the construals be distinguished, moreover; further distinctions are required within each one. Thus the notion of information processing—responsible for such slogans as *The Information Age*, and the link between philosophy of computing and philosophy of information—must be broken down into at least three sub-readings, depending on how *information* is understood: (i) as a lay notion, dating from perhaps the 19th century, of an abstract, publicly-accessible commodity carrying a degree of autonomous authority; (ii) so-called "information theory," the semantics-free notion originating with Shannon & Weaver (1949), which spread out through much of cybernetics and communication theory, is implicated in Kolmogorov and other complexity measures, and has been tied to notions of energy and entropy; and (iii) the semantical notion of information advocated by Dretske (1981), Barwise & Perry (1983), Halpern (1987), and others.

The second involves a set of 4 "dialectics"—fundamental metaphysical distinctions particularly applicable to "things computational," and necessary to understand if we are to claim to have an intellectual grasp on computing.

1 • **Meaning and mechanism:** The first dialectic involves the only substantial thesis about the nature of computing I adopt as an investigative guide (again, not as necessarily true of the subject matter, but indicative of issues to be investigated): that, in one way or other, computation involves an interaction or interplay of *meaning* and *mechanism*. That computation is somehow mechanical is reflected in the fundamental effectiveness limits that permeate computational theory and experience. As already suggested, there is disagreement about the nature or origin of this "efficacy"—whether it is (i) an abstract notion, as gestured towards in the notion of an "effectively computable function," taken by logicians and mathematicians to be an entirely abstract notion, unrelated to physical constraint; or (ii) a physical notion, tied to underlying physical law. But as so powerfully demonstrated by Turing in his original paper, that computation is in one way or another limited both in principle and in practice is as deep a fact about the topic as any that exists.[21]

   That computation has anything to do with meaning, interpretation, semantics, etc., is much less widely agreed—in spite of the use of logical language discussed above. I take the semantic nature of computing to be compelling, however, both from the nature of existing theoretical debate and from the character of the phenomenon.

2 • **Abstract and concrete:** A second distinction that permeates computing, which has arisen several times already, is that between the *concrete* and the *abstract*. What "degree of concreteness" computation manifests, if I can put the question that way, is deucedly difficult to figure out—to say nothing of what, under scrutiny, the terms even mean. Are arrangements of physical things themselves physical, abstract, or somewhere in between? ("I like what you've done with your living room; that's a *great arrangement of chairs*.") What about abstractly specified concrete properties, or concretely specified abstract properties? Or do the words signify neither a binary distinction, nor two ends of a continuum, but some

[21]Students think Turing is famous because he introduced the notion of a computer, and demonstrated its power. It is important to remind them that he demonstrated *both its power and its limitation*.

third possibility entirely? Perhaps they aren't even the right contrast pair. Only the philosophy of computing knows for sure.

3 • **Static and dynamic**: Less philosophically vexed, but as crucial to computing, is the distinction between *static* and *dynamic*. Programs, it would seem, are static entities, or anyway passive;[22] compilers translate them into other static entities (programs in a lower-level language); interpreters "run them," generating dynamic processes, etc. Or rather: interpreters are programs too; it is when interpreters *run* that they take programs and generate further process or behaviour—behaviour somehow different from, and yet in other ways coincident with, the behaviour of the interpreter's own running.

Some immediate facts aren't hard to delineate, in other words—even if the saying gets pedantic. Still, the distinction is important, as is the question of whether the way we currently arrange things is necessary or mere historical contingency. Is it just habit, or lack of imagination, that makes us think process specification should take static form? Could a dynamic process itself describe, represent, or specify?[23] If so, surely there could be computational analogues, suggesting that we shouldn't build static specification into our framework. In this and other cases, it is clearly important, in so far as it is possible, to avoid shackling our philosophy of computing to the tiny fraction of possible computational architectures that have so far been explored.

4 • **One and many:** Finally, any account of computing worth its salt must deal with a bewildering plethora of distinctions between "things that are one" and "things that are many," such as a single program, web page, file, etc., and multiple distributed "copies" or "versions" of it (a distinction that bedevils software projects and replicated data bases), or the issues that arise when you call a procedure on a matrix: do you pass a distinct copy or, as it is said, "the address," so that there is only one—or is that rather a new copy of "the same address," i.e., two pointers (copies?) that point to the same location, or… Attempts at ultimate clarity can lead to madness.[24]

[22]This is not to say that people don't update them—i.e., make better versions *of the same program*; the identity conditions are complex, but programs certainly exist over time.

[23]Ask a friend to describe a spiral staircase, and watch their hands; you will see a dynamic representation.

[24]It is uncanny how sophisticated expert programmers are at navigating these singular/plural

We speak of many/one relations in many different ways: (i) in terms of *types, classes, categories, templates, patterns, schemata,* etc., where a single (abstract?) entity is taken to have multiple *instances*; (ii) as a (concrete?) unit thing with different *copies, editions,* or *versions*; (iii) as a *set* with distinct *members*; (iv) as a *role* played by different *individuals*; and so on. It is far from clear that we understand the distinctions between and among these ways of speaking, and why, exactly, we use one or other in any given case. More seriously, the profusion of possibilities, and the diabolical fact that on reflection whether something is "one" or "many" can seem to be a matter of perspective or even degree, rather than being an intrinsic property (of it? them?), can send metaphysical tremors through the foundations.

Another issue on which to keep an eagle eye.

3c — Formality

The third axis has to do with **formality**—one of the most recalcitrant properties underlying the entire field. Somehow or other, it is thought, computation is a *formal* phenomenon, or amenable to *formal* analysis, or works *formally,* or something like that. Just which of these is true, what they mean, and how they relate, are additional issues that any philosophical analysis of computing must investigate.

The near-universal allegiance to formality is both curious and fertile. It is not as if "formal" is a technical or theory-internal predicate, after all—no one writes FORMAL(x) in their equations. Moreover, informal usage seems to range across as many as a dozen meanings of the term: *precise, abstract, syntactic, mathematical, explicit, digital, a-contextual, non-semantic,* etc. Far from engendering debate, this profusion or outright ambiguity has probably helped to cement consensus. Because it remains tacit, cuts deep, has important historical roots, and permeates practice, formality is an ideal foil with which to investigate computation.

Once again I will cut straight to the bottom line. The moral for computer and cognitive science here is similar to the claim made earlier about the seven construals: *no plausible reading of 'formal,'* in my view, *applies to the computational case.* Needless to say, negative claims are tricky to prove. To make such a conclusion watertight, one would need both an agreed theory of computing and

shoals.

a definitive analysis of 'formality.' But certainly my investigations have led me to conclude that there is no substantive reading of 'formal' under which concrete, in-the-world computing—*computation in the wild*, as I sometimes call it—is, in fact, necessarily formal. As I put it in another context, "one cannot avoid the ultimately ironic conclusion: that the computer, darling child of the formal tradition, outstrips the bounds of the very tradition that gave rise to it."

4 • Results

The issues discussed above are given slightly more treatment in Smith (■■), and will be explored in depth in Smith (forthcoming). Here, though, it is time to assemble these piecemeal results into the ultimate conclusion, and sketch some of the issues it opens up in front of us.

The bottom line again is simple. Not only do none of the seven construals, understood formally or informally, serve as an adequate account of computing. More seriously, no other construal, of my own or anyone else's making, will serve either. The reason is stark: *there is no theory of computing to be had*. This, too, is a result that after this long journey I am prepared to claim: the term 'computational' does not name a property of theoretical significance.[25] A philosopher who believed in such things might say that *computation is not a natural kind*, though not being such a philosopher, that is not how I would put it. I would rather just say this: that there is ultimately nothing *special* about computing or computers—nothing to give substance to a theoretical notion of computing—beyond the thesis of the first dialectic: computers are systems or devices that involve an interplay of meaning and mechanism, the best we know how to build. Period.

"All great ideas languish for most of eternity in obscurity, have one brief moment of glory, and live out their dying days as a platitude."[26]

There is nothing more to say.

The first comment to make—and it should be made straight away—is that this is wildly optimistic claim. Far from being negative, the fact that there is no theoretical substance to something's being computational (i) not only opens up the realm of computing to possibilities not heretofore imagined, but (ii) from an

[25]Or *computing*, or *computer*; it doesn't matter.

[26]I heard this saying, growing up, from my late father; whether he had heard or created it I do not know.

intellectual point of view, makes the development of computers vastly more significant than it would otherwise have been. Sure enough, a number of popular hypotheses end up on the cutting room floor—including the vaunted computational theory of mind.[27] On the other hand, it follows that all of the specific details and understandings and intricacies and mechanisms and architectures developed in computer science are "unrestricted": rather than applying to just a *subset* of the world's systems, they apply to *all* systems—at least all that involve the fundamental meaning/mechanism dialectic, which is a lot. So to take just one example: the relation discussed above between philosophy's notions of reduction, supervenience, etc., and computer science's understanding of architecture, implementation, abstraction boundaries, etc., are not just *parallel* developments. First blush was right: they are theoretical perspectives on *the same subject matter*. That is why that doctoral dissertation would be important—a synthesis of the two perspectives is mandated by the simple but compelling fact that the subject matters do in fact coincide.

I am not saying that the development of computing is not a theoretical (as well as practical) accomplishment of the utmost magnitude. The discovery of how to arrange physical matter in such a way as to implement digital processes, for example, is a staggering achievement—easily worth a passel of Nobel prizes. Rather, the point is that, instead of being viewed as a restricted species, as is implicit in the idea that 'computational' is a property of theoretical significance, computers are better understood as a *site*—a "laboratory of middling complexity," where we can work out the best understandings we can muster about how meaning and mechanism interact.

5 • "Internal" Prospects

What lies ahead?

For discussion purposes, I will address this question using a distinction that is at the very least not black-and-white, and ultimately not one I believe in at all, but which will nevertheless bring some order to the discussion. I will divide my remarks into two categories: (i) "internal" prospects for working out the theoretical and scientific consequences of the views to which I argue, and (ii) more profound "external" implications, regarding our fundamental approach to metaphysics, ontology, and epistemology.

[27]Of course we are computers (unless substance dualism is true). We are physical beings, and we mean, or deal with meanings.

5a · Internal

I said that computing involved a mixture of meaning and mechanism.[28] I also said that computer science uses a spate of terms borrowed from logic—including some (*identifier*, *symbol*, *reference*, *interpretation*, etc.) that, in logic, have squarely to do with semantics. It would be natural to suppose that these terms are used to describe the semantic or "meaningful" aspect of computing, rather than the "mechanism" side. Perversely, however, the converse turns out to be true. This is one reason why both Searle and his interpreters get confused. After recruiting semantical concepts (or at least terminology) from logic—terms or concepts that logic uses to analyse meaning—*computer science deployed them to study additional aspects of mechanism.*

How this came to pass is a complex story, but the result can be roughly caricatured. Computer science needed to understand the relation between a **program**, taken as a static or anyway passive entity that, plus or minus, both *describes* and *prescribes* a "computation," which for present purposes we can take to be the dynamic **process** that takes place when the program, as we put it, "runs." Because of the descriptive element, it was easy to parlay logic's notion of semantics to this purpose, since it took the form of mapping between one thing, "syntactic" or "grammatical" in form, and another, which logic had analysed in terms of a mathematical model. Logic's "semantic interpretation function" could thus be used, in computer science, to map programs onto the resulting processes, mathematically modeled or abstractly described. Some work needed to be done. In order to capture the *prescriptive* part, for example, the interpretation relation needed to be constrained to be *effective*—in a way that would be perverse, if not outright unimaginable, in classical logic. From computer science's point of view, however, the move made sense. It is this restriction of interpretation to effectiveness that has spawned computer science's obsession with constructive mathematics, intuitionistic type theory, and eventually the development of linear logic, all in service of a kind of ultimately concretized meaning.

The problem with all this, however, is that the genuine semantical question, from a philosophical point of view, is not about the relation between program and process, but between process and world—between NASA's system to calculate trajectories and the *orbits of distant planets*, for example, or between the

[28]Admittedly, I haven't defended this statement—merely assumed it as the first dialectic. Given what is said in this and the next two paragraphs, it is not as simple a thesis to defend as one might expect, and so I will continue to do so here. See Smith (forthcoming).

proximal results of NATO's early warning systems and the distal fact of whether an intercontinental missile strike *has in fact been launched*. Schematically, that is, we are faced with two relations connecting three realms: (i) between a program P and the process R that results from running it; and (ii) between that process R and the task domain D that the process is about, or that is the subject matter of the information that the process manipulates, or whatever. Computer science has used logic's semantical vocabulary to study the first relation, $P \Rightarrow R$, whereas the tough semantical question is about the second relation, $R \Rightarrow D$.[29] That remains to be theorized. (It is also a tough point to make to computer scientists, since you can't use any of logic's classic semantical concepts to describe it, as they have already all been "used up.")

One more technical result needs to be brought out. Earlier I mentioned debates about the "origin" of the computability limits first demonstrated by Turing, which form the foundations of computability and complexity theory. One of the results that emerges from the analysis of the effective computability (EC) construal is that recursion theory, the notion of computability, etc., turn out to be *mathematical models of physical constraint*. So that theory, too, is about the "mechanism" side of the substantive dialectic. It is framed as if it were a theory about the computation of numbers, but in fact it is a mathematical theory about reconfigurations of marks (i.e., of physically distinguishable states). This is obvious to programmers, increasingly recognized by computer scientists, and anathema to logicians and recursion theorists. But no matter; it is again a tremendously positive result. The development of the theory of effective computability, once reconstructed as a mathematical theory of causation (below), is another intellectual achievement worthy of several trips to Oslo.

Given these understandings, I would identify the following four projects as the first half of my answer to Floridi's last question, about the most important issues facing the philosophy of computing. As I say, I see these four as "internal" to the subject matter. Even they are huge, and barely begun—but that just underlines what I said above: this is a new field, with most of the work remaining in front of us.

5a · 1 — Concretisation

If computability and complexity theory is about mechanism and process, not numbers (cf. C4, above), then the entire theory must be recast in concrete terms.

[29]As I once put it to Gordon Plotkin, a programming language semanticist, "I am interested in the semantics of the semantics of programs."

To take just one example, consider the infamous equivalence proofs discussed earlier. As currently cast, they claim that, if set up with appropriate inputs, one machine $\boldsymbol{m}_1$ can "do the same thing"—that is, can "compute the same function"—as another $\boldsymbol{m}_2$, if given the same input. As I said, that statement relies on a notion of "computing," which for two reasons must now be rejected. First, since 'compute' can no longer figure as a substantial property, we need to cleanse all theoretical statements of its use. Second, to the extent that there was any meaning to the phrasing "machine $\boldsymbol{m}$ computes function $\boldsymbol{f}$," it is this: given "input" marks $\boldsymbol{j}$ denoting $\boldsymbol{x}$, machine $\boldsymbol{m}$ can produce "output" marks $\boldsymbol{k}$ denoting $\boldsymbol{y=f(x)}$. "Computing a function" has to do with mathematical entities—that is, with $\boldsymbol{f}$, $\boldsymbol{x}$, and $\boldsymbol{y}$. On the recommended concrete overhaul, the theory would have to eschew all mention of such entities, and speak instead about machines and marks: $\boldsymbol{m}$, $\boldsymbol{j}$, and $\boldsymbol{k}$.[30]

This transformation will be massive. Just one example: It is widely understood, and used to encrypt credit card numbers on the internet, that "factoring the products of two large primes" is a difficult task. But factoring is a phenomenon in the realm of *numbers*, not in the realm of *physical arrangements*. Moreover, factoring numbers is trivial if numbers are represented in non-standard ways—for example, as lists of their prime factors. So whatever is going on, "what is hard" must have to do with the nature of *numerals*. As such, it deserves framing in terms of marks, directly.[31]

I dub the recommended reconfiguration of the equivalence proof a **motor theorem**, with roughly the following content: Given a motor $\boldsymbol{m}_1$, and an adequate stock of other passive but perfect parts,[32] one can assemble a configuration $\boldsymbol{p}$ of those parts, such that the resulting device, consisting of $\boldsymbol{m}_1$ appropriately connected up to $\boldsymbol{p}$—a device of potentially Rube-Goldberg complexity—will produce "isomorphic" behaviour, under a hugely broad metric of "equivalence,"

[30]The theory might (though I don't know whether it will) use functions and numbers to model or measure the concrete phenomena, in the way that physics uses numbers as a basis of measurement; but the resulting theory will no more be *about* numbers than to say that earth's escape velocity is 11,200 meters per second is a statement about the number 11,200.

[31]What kind of fact is that? It must have something to do with the composition of prime factorization with the inverse of the interpretation function for radix numerals—or rather that composition (or something like it) may play a role in its mathematical characterisation. But what that comes to, concretely—and why such an operation should be hard for a mechanism subject to our physical laws to perform—is going to take some work to figure out.

[32]Friction-free, totally discrete, and idealized in various other ways—these are the consequences or strictures of digitality, perhaps the most significant notion in the entire computational pantheon.

to that of any other machine m_2 that can be built.

Is the motor theorem impressive? Should we be impressed that such a theorem can be proved? Who knows? Personally, I should not have thought so.[33] Most concrete devices consist of some number of motors, gears, pulleys, containers, pipes, etc., or other forms of motive force. It does not seem to me especially odd that with one motor, of sufficient (that is: indeterminate) size, plus an indefinitely large supply of other perfect, friction-free parts (switches, ropes, pulleys, etc.) one could construct a device functionally isomorphic to any "perfect" device—especially if the metric of equivalence that one is mandated to meet, as in this case, is sufficiently broad. But normative assessment may be personal, and anyway should wait until much more of the reconstruction worked out in detail—something not yet done. The present point is simply that something like the motor theorem, plus appropriately detailed variants for all of the complexity variants, is mandated by the concrete understanding of the notion of "universality" that comes out of this analysis.

In passing, one salutary effect of this "concretization" of our understanding of computation may be to help rid popular culture of various myths about computing, including the ubiquitous belief (false, in my view) that there is a fundamental distinction between the "virtual" world, on the one hand, and the "physical" or "real" one, on the other. Not only are computational processes (and worlds) *real*; they enjoy a materiality that, while different in tenor than that of our direct experience, is undergirded by the same physical laws and participates in the same temporality.[34]

5a · 2 — Physical states

Another issue brought onto centre stage by the concrete reformulation of the mechanism side of computing is that of individuating physical states (or perhaps more correctly: physical state types). As Putnam points out,[35] one can claim that a rock implements any computation one wishes so long as one divvies

[33]I could never understand, when I first learned about the proofs of universal computability, why, in spite of their surface brilliance, I found them, *au fond*, to be so fundamentally boring. It took almost 20 years to figure out the answer.
[34]Curiously, given contemporary processor speeds, the material worlds of computing manifest its relativistic character quite directly. It matters, if you are a thread running on a contemporary processor, that a nanosecond is approximately "equal" to a foot—in a way that doesn't enter our everyday phenomenology.
[35]«reference: the appendix to *Representation and Reality*»

up the physical states "appropriately"—which is to say, in completely unnatural ways. Deviant physical typing can produce lots of strange results: solving the halting problem, decrypting the most challenging encodings, solving traveling salesman problems in unit time. Of course, such Goodmanesque predicates[36] violate both intuition and utility. But what *is* an appropriate physical state? No one knows. All we can say is that an adequate theory of meaning and mechanism depends critically on the answer.

5a · 3 — Process

"Computation," it used to be said, "is mathematics plus time." I disagree with the mathematics part, but the inclusion of temporality as computationally fundamental is unarguable. Process, doing things, what can happen, how long it takes—these are constitutive of the computational realm. Yet I think it is fair to say, almost a century after Husserl, Whitehead, and Heidegger, that we still do not have a theory of process and temporality worthy of the name. The temporally dependent variables of physics, enshrined in the calculus, are one spectacular success story, but they are extremely specific. There is no reason to believe that the "soul of a meaningful machine" will be disclosed through numerically valued measure properties.[37] By and large, computer science doesn't use them, instead analysing dynamic systems in terms of static structures—programs, inputs, outputs, requirements to be met, conditions to be honoured, contexts viewed as static abstractions. Why don't we deal with time more directly? As I continually ask graduate students, where is the programming language that is as natural for expressing jazz rhythms as Lisp is natural for expressing recursive functions? Should such a language *itself* be dynamic? Even in cases where we do employ mathematics, process and dynamics are modeled in terms of abstract a-temporal structures. Is the atemporality of mathematics a metaphysical, epistemological, or cognitive necessity? Is a dynamic mathematics[38] an unthinkable possibility?

History, I suspect, will laugh at us three times over: once for our reliance on

[36]Such as Goodman's famous *grue* and *bleen*: green before some time ***t***, and blue thereafter; and its converse.

[37]Experience with computing to date suggests that architecture is a deeper analytic concept than measurement.

[38]Not just dynamic notation, nor a dynamic system about mathematics, nor a dynamic system mathematically modeled, but dynamic mathematics itself—such as sets changing membership over time, a hole opening up in a topological manifold, or an Abelian group's multiplicative operator losing its commutativity.

objects, twice for the skittishness with which we approach relations, and three times for our naïveté about time.

5a · 4 — Semantics

Finally, and most obviously, we need a theory of semantics—of the "meaning" side of that first substantive dialectic. In spite of the fact that essentially all of contemporary computer science's theoretical apparatus deals only with the mechanism side, it is not pure physicality that we are up against,[39] but, as I keep saying, meaningful mechanisms. Cognitive science recognizes the problem, e.g., in the so-called "symbol grounding" problem. But for an adequate intellectual understanding of semantics, intentionality, meaning, interpretation, reference, modelling, analysis, simulation, etc., we remain woefully in the dark.

It is on the semantic side of the equation that some look to the notion of information. The question is whether the "counterfactual correlation" analysis of information content inaugurated by Dretske in 1981,[40] or perhaps the teleosemantic variants developed since then, could be pressed into service for a genuinely semantic account of what information is, on which, in turn, a semantically grounded information-processing (IP) construal of computing could then rest. There are huge challenges to these accounts, involving such issues as how to deal with "misinformation" (if that is a species of information at all), avoid various forms of pan-informationalism, etc., but the effort is arguably the only substantial idea in town as to what a semantical account would actually look like.

My difficulty stems from two sources. First, as I will discuss more in a moment, I believe that extant accounts of information are fatally dependent on undischarged ontological assumptions, and therefore cannot serve as a basis for a thorough-going philosophical investigation. Second, however, and more immediately pertinently, I claimed that not all computing can be understood as "information processing," and therefore that the IP construal will not work as a general analysis (semantical or not). In a nutshell, the problem is that not all computing is *about* something else, as the notion of information would imply, but actually deals with things *in themselves*. When my email client tells me that I have new email, it does so by representing that fact (in English) in a window on my screen. When I go to retrieve the email, however, the computer *actually delivers it to me*; it does not merely provide me *information* about it. Should your

[39]Contra such writers as Phil Agre (■■), who argue that computing is merely a practice of building physical stuff.
[40]*Knowledge and the Flow of Information*, 1981.

message fail to arrive, it is not that I did not receive *information* about your communication. Rather, what is true is exactly what I say: "I did not *receive* it." Information-processing is not a strong or general enough notion to deal with the genuine encounter and engagement in our lives that computers and other meaningful mechanisms (like people) manifestly exhibit.

And so while I am as supportive as anyone of pursuing work in the philosophy of information,[41] and even teach a graduate seminar on the topic, I believe the vast reaches of an encompassing and adequate theory of the semantic dimension of meaningful mechanisms remain largely unexplored.

6 • "External" Prospects

Things are serious when (i) the "mechanism" side still needs theorizing, but we lack an account of physical types on which to base it; and (ii) the "meaning" side remains almost wholly unreconstructed. You might also think, since I have spent all these years struggling with the issues, that I would have something positive along these lines to propose.

In a way I do, but at best a cursory sketch. A hint of the reason is contained in the fact that we lack a theory of physical typing. After taking this long journey through the computational landscape, the most sobering result of all is to realise that the most serious problems to be addressed, in developing a theory of meaningful mechanisms, are *ontological,* not just mechanical or semantical. Even more seriously, the ontological issues involve an *inextricable mix* of mechanical and semantical concerns. Ontology, that is—by which I mean an ontological account of the entities necessary in order to give an account of computing and other meaningful mechanical systems—is inexorably tied into semantical or intentional or epistemological issues of meaning.

Space prohibits any real defense of this conclusion here, which is anyway too consequential to accept lightly—though the complexity of the subject matter revealed by a close look at all four dialectics may suggest some of the reasons. The magnitude of the impact, however, is not hard to see. Within traditional science and analytic philosophy, it is traditional to accept the following "division of labour": (i) to assume that the "ingredients" out of which an account will be constructed can be distinguished and identified in advance: the objects and properties and relations and sets and states of affairs and so forth; and then (ii)

[41]Which is not to say that I believe that a theory of information is there to be found. We'll have to see.

to develop the account of the meaning or semantics or epistemology in terms of them. Ontology, that is, is not only assumed to be separable from epistemology, but to precede it, in some logical or metaphysical sense. For example, if you were to write a computer program to control an elevator, you would first specify the world of elevators, floors, passengers, buttons, cables, etc., and then write the program *in such terms*. Requirements engineering pretty much assumes this.

The conclusion I have come to is that this approach will not work in the long run. As argued by legions of philosophers of a more literary stripe, ontology (what the world is like, in any intelligible sense) and epistemology (how we take the world to be) need to be reconstructed together. If we use 'metaphysics' to name that conjoined effort, then the answer to the original question about developing an adequate account of computing—i.e., as we can now see, a comprehensive theory of the meaning/mechanism dialectic—involves nothing less than a full-fledged assault on constructing an appropriate metaphysics.

So be it. For a bit more discussion, see Smith [■■]; and for an inchoate stab at what such a metaphysics might look like, Smith [1996].[42]

In a way, the conclusion isn't surprising. Think about persistent online worlds—and the vexed questions that come up about whether avatars, into which people pour thousands of hours of devoted labour, are: (i) prostheses, (ii) beings in another reality, (iii) names or representations, (iv) identical to the player (warranting the common use of the term 'I' in such phrases as "I am going to slay the demon"), etc. Is a "crime" committed in such a world as innocuous as those investigated by Hercule Poirot, as serious as a "real-life" version, or something in between? And if the last, what existential or ontological conception of what is going on is strong enough to found such a ethical regime?[43]

What is striking is that ontological challenges aren't just "out at the level of use"—i.e., where people manifestly enter the picture. They permeate the entire subject matter. Even accounting for the identity conditions on a file outstrip the capacities of any known account. Is/are the file in the file cache, the one on the backup tape, the one I sent to you by email, the same? Or a copy? Sometimes it is convenient to think of it one way, sometimes the other. But is identity dependent on how we *think* of it? Maybe—but that's no innocent complication.

[42]Note too the extent to this conclusion, wrung from an allegedly technical subject matter, resonates with claims made in feminist epistemology, science studies, and other poststructuralist initiatives. Not evidence for anything, exactly; but not sheer coincidence, either.
[43]See Kevin Eldred's forthcoming doctoral dissertation for an in-depth analysis.

Similarly, any simple distinction between a sign and what is signified (name/named, description/described) is too blunt an instrument to deal with even simple computational systems. Does the ASCII version of a visual program *represent* the program, or *is it* the program, or is it a *translation* of the program? And what about the file I thought I lost, last night—but then realised that I didn't, because I had made a backup just a few hours before. Sure, I lost a few hours of editing—but still, I found "it." Says who? Says I. Which means that "the file," for me, is a singular term referring not to a particular physical copy, or even to a simple type of physical copy, or perhaps even to a more abstract single individual that the physical copy "realizes" (what's the difference between those two ways of putting it, anyway?), but to something yet more abstract—something whose identity conditions are more like the identity conditions on proofs we rely on to decide whether a young mathematician should be awarded tenure for their discovery of a "new proof" of a known result. In both cases, I would hazard, identity cannot be established independent of meaning—and perhaps even purpose. A sensible enough claim—but again a seriously expensive metaphysical result.

The problem, of course, is that once this gate is opened, and we take a step through, we enter a terrain of virtually unlimited grandeur and scope. The foundations of a great deal of what we consider science fall away, replaced by metaphysical and epistemological questions of almost unutterable consequence—and, needless to say, surpassing difficulty.

Objects, and in fact all of commonsense ontology, need naturalising, for starters—as much as any semantical or intentional notion. It isn't obvious where semantical or intentional notions will come from, either, since there won't yet be any stable ontology on top of which to build them. The semantic notion of information, for example, won't be able to speak of counterfactual dependencies between entities, at least not if it is going to play a foundational role on which those entities are going to depend. Or perhaps objects and information will arise together, with objects being patches of the world understood or "parsed" ("coarse-grained," as AI would put it) at a level or degree of abstraction that facilitates the kinds of counterfactual correlation that in turn allows us to track them. Who knows? It is not a crazy idea, even if how one would make good on it is not exactly obvious. It isn't just objects we need, either; the same goes for properties, relations, sets, etc.—to say nothing of "truth-makers" for non-conceptual content, be that Strawsonian feature-placings or whatever. Norms, too, or something to fill their role (perhaps fundamentally dynamic?) should be added to the list.

And so on. All I want to emphasise here, however, is the role of computation in this vast enterprise. For many years metaphysics has been viewed with huge suspicion—one of the few things on which both modernists and postmodernists agree. I am claiming, in contrast, that we are not going to understand computing—or meaning and mechanism more generally—unless and until we get over that suspicion, and take up the metaphysical gauntlet for real. Crucially, as I will argue elsewhere, that does not mean we need fall prey to any of the ways of doing metaphysics that have convinced a few centuries of philosophers that it is a hopeless and hapless enterprise. Interestingly, moreover, but consequentially, and something else that will need careful explanation, I believe that we can do so *empirically,* using computation as our laboratory—and not just metaphysics, but an indissoluble mixture of metaphysics, ontology, and epistemology; and not just theoretically, either, from an armchair or with Platonic detachment, but in an engaged, constructive, probably quite messy and concrete way.

Computers are not a subject matter, but as I said above, *laboratories of middling complexity*—vastly more complex than the atoms and frictionless pucks and pendula of simple mechanics, but vastly simpler than anything even reminiscent of human cognition. Whereas I identified four major challenges for future research "internal" to the study of computation, this is the one challenge—or rather, opportunity—I would name from the more serious and more sobering external perspective: that we recognize the first hundred years of computing as something of an Alchemical precursor to the intentional or meaningful sciences, and, with unswerving focus, parlay our computational experience into a finally successful metaphysics.

One final point, to bring the story full circle. The term 'material' is the adjectival form of 'matter' in both of its senses: 'matter' as a noun, as in "slurries are a form of matter studied by geologists"; and 'matter' as a verb, as in "it doesn't matter whether you call me or not." When we speak of *material objects*, most people assume we are using the form derived from the noun—that a *material object* is something that weighs something, that occupies space, that you might bump into. A material argument, however, of the sort a judge might deny you had raised, is of the other kind: an argument that *doesn't matter* (to whatever issue is at hand). How the two forms of 'matter' came apart could be argued, but suppose we lay it on Descartes. Then one way to describe the project laid out above is that of developing an understanding of a material object as a "spatio-temporal chunk of reality that matters"—thereby healing a temporary 300-year rift between *matter* and *mattering*.

And with that we can finally answer the questions with which I started. Can studying computing help us do to the richness and complexity of the human condition? Yes, but not in the way that I thought, back then. Computers can help by serving as a laboratory in terms of which to explore issues of intentionality, embodiment, semantics, meaning, mechanism, interpretation, etc., so long as we let go of the conceit that they are computers—or anyway, the conceit that that their being computers is theoretically relevant. Can they be understood with all the power and insight and elegance of the sciences? Well, no—not if elegance requires formality. But formality has lost its sheen, for me at least, and I find more reward in exploring the metaphysical depths that these seemingly innocent devices have opened up in front of us. So yes, in a more grown up way—a way I just wasn't up to, at the time. Finally, as an added bonus, the time wasn't wholly "off" from physics, after all—as maybe my professor knew, and anyway is betrayed in the etymology. Maybe metaphysics is just physics, pushed harder. Hard enough to unleash meaning.

Time for second semester.