Introduction[1]

When physicists bring fundamental particles into contact, the process is not exactly mild. Only at very high energies do particles genuinely interact—and thereby reveal their secrets. The encounter is violent, and the disclosure profound. Increasingly, moreover, it is on the basis of those hard-won secrets, wrung from destruction, that physics' vaunted theories are founded.

My view of computing and philosophy is similar. Powerful secrets will emerge from their encounter—but only if they collide at high energy, and at peril. To date, in my view, the interaction has been too gentle. Intersecting at moderate energies, the fields have mostly bounced off each other—adjusting their external trajectories, but largely remaining internally intact. Nothing drastic enough has happened to undermine their identities, reconfigure their role in the intellectual pantheon, or—what matters most for my purposes—force either to expose its innermost mysteries. As a result, our eyes are not yet open to some of the most fundamental insights on which the future of both will depend.

1 Engagement

It is not that computer science and philosophy are unacquainted. Computing grew out of logic, metamathematics, and other philosophical projects, for starters—a fact still betrayed in its technical vocabulary.[2] Ever since the formal development of

1. Version 0.73 — February 20, 2012

2. Consider a sampling of computational notions: *symbol*, *reference*, *identifier*, *language*, *syntax*, *semantics*, *evaluation*, *interpretation*, etc. In their technical use, these terms not only derive from logic and the philosophy of language; they all also implicate the 'meaning' side of 'meaning/mechanism' dialectic. They are semiotic, normative, or what philoso-

computation almost a century ago, moreover, philosophical treatises have been written on computation and computability, logical frameworks used to analyse the semantics of a wide range of computational system, and investigations conducted on the intersections of programming and provability. Methodological papers have also been published, documenting the impact of computational practice on the philosophical enterprise, electronic journals and web-based resources set up to facilitate philosophical discussion,[3] etc. And substantively, in what is at least arguably the most intense encounter, theorists from both camps have expended effort exploring the idea that we ourselves might be computers—i.e., investigating what is widely known as the *computational theory of mind* (CTOM).

Within these broad outlines, the specific intellectual profile of discussion has varied, over time, as different topics have moved into and receded from prominence. Initially, particularly in the 1930s and 1940s, philosophical discussions of computation were primarily conducted in the context of logic and the philosophy of mathematics—e.g., in the work of Gödel, Turing, Kleene, Gandy, and others. In the second half of the century, after the 1950 publication of Turing's "Computing Machinery and Intelligence," and the emergence of artificial intelligence (AI) and cognitive science, most philosophical analyses emerged instead from within the philosophy of mind—in for example the work of Dennett, Searle, Haugeland, and Fodor.

Even as the focus of philosophical engagement shifted from mathematics to cognition, the originating connection with logic remained in view. Logic's influence on the CTOM was evident in discussions of the appropriateness of a logical

phers would call *intentional* notions, that is—a striking fact for a field that many view as representing the triumph of the age of mechanism. In connotation as well as meaning they are markedly different from such physical notions as *cause*, *force*, *locality*, *mass*, *impulse*, *impact*, *energy*, *momentum*, and so on, as one might otherwise have expected for a project so interested in *machines*.

3. «Ref http://philpapers.org, and bbs»

or logicist conception of cognition, for example—the idea on which the first generation of computational theories of mind were founded, amd the conception of artificial intelligence immortalized in Haugeland's label "good old-fashioned artificial intelligence" or GOFAI. This approach continued to attract attention, in the face of increasingly strong opposing views, throughout the 1980s. As regards the trio *logic*, *computing*, and *mind*, in fact, it was widely assumed (though never by me) that the first two, *logic* and *computing*, were intrinsically and inextricably associated, and thus that to the extent that mind was seen as *not* appropriately understood in terms of logic and rational inference, then to that extent mind *must not be computational, either*. Such presuppositions about the intrinsically logical character of computing played a large role in arguments throughout the 1990s for a shift from so-called "computational" models of intelligence[4] to a rash of dynamic, networked, neuroscientific, situated, and social alternatives (cf. van Gelder, Kelso, Edelman, Brooks, Suchman, and others).[5]

During the same period, though with a very different character, a more direct engagement between logic and computer science developed which, rather than focusing on overarching epistemological questions about the nature of knowing, took up more detailed formal considerations of proof theory, the "semantics"[6] of programs, and the place of constructive and intuitionist mathematics in theoretical computer science (cf.

4. "I say "so-called 'computational' models" because it is far from clear that connectionist, dynamic, situated, and other networked proposals are in fact noncomputational. In fact I myself do not believe it. It is striking, and far from irrelevant, that those who most strenuously argue against computational models under this logicist conception of computing, such as van Gelder in philosophy and Edelman in medicine and physiology («refs?»), are not computer scientists.

5. «Appropriate refs». These arguments would have been more forceful and to the point, in my view, had they been framed as being opposing to a "logicist" models of intelligence.

6. 'Semantics' in quotes for reasons explored in … «■■».

Scott, Martin-Löf, Constable[7]).

Recently, a third area of mutual interest has come onto the scene. Rather than vieweing each other indirectly through the lenses of logic or mind, computing and philosophy have once again rubbed up against each other, this time through a common interest in the notion of *information*. Whether the term 'information' names a common or even allied subject matter in the various discourses in which it is increasingly theorized remains a matter of debate (information is now a technical term in at least economics, biology, physics, communication, sociology, and media theory, as well as computing and philosophy). Nevertheless, it is undeniable that, over the last decade or so, the philosophy of information has risen sharply in theoretical visibility, in part due to the targeted efforts of Luciano Floridi.

Even if information joins logic and mind as a topic of common interest to both computer science and philosophy, the interaction between the two fields remains to that extent indirect—as well, in my view, as too modest. I find it especially striking that, in spite of the undeniable impact of computational ideas on the world, ourselves, and our place within it—i.e., in spite of the fact that we are unarguably living through the development of a science and technology of immense ontological and epistemological impact—the *philosophy of computation* is not (yet?) a recognized area of specialization within the philosophy of science. I know of only a handful of philosophers, world-wide, who list "philosophy of computing" on their CVs—a situation vastly different from that in the philosophy of mathematics, philosophy of physics, or, over the last few decades, philosophy of biology.[8]

---

7. «ref, and comment on the title of, such papers as "Programs as Proofs," "Proofs as Programs," etc.»

8. For reasons touched on below, I have reasons to doubt that a project directly focused on computation should be treated as within the philosophy of *science*—and more seriously, even that it would constitute a legitimately delineated area of intellectual inquiry at all. But those concerns are primarily intellectual, rather than political or historical. I

* * *

If philosophy's interest in computing has been modest, computing's interest in philosophy has been even more limited. When the interaction between the two fields is viewed from the computational side, it is even more striking how much logic has been the primary subject matter. Even AI, which one might innocently imagine would have focused primarily on philosophy of science (e.g., on theories of rational belief revision in the face of accumulating and sometimes contradictory evidence) if it was going to turn to philosophy for inspiration at all, or even on psychoanalytic theory if it were to cast its view more widely, has largely restricted its attention to a relatively traditional, rationalistic conception of logic.

But the strongest point of contact has been in the realm of programs and programming languages—both substantively, at the level of the programming language design itself, and also metatheoretically, in the use of logical frameworks for analytic

still believe it is perplexing, and mostly unfortunate, how little direct philosophical treatment a development of the magnitude of computing has received in philosophical circles.

Note, too, that although there are some who would ground computing on a notion of information, and although as well there are those who construe computing as "information processing" «ref AOS», the thesis that we are computers is still called the *computational* theory of mind, not the *informational* theory of mind. And the information-based accounts of mental semantics, such as those based on Stamp, Dretske, etc., many of which subsequently morphed into teleo-informational versions through reliance on evolutionarily based notions of biological function to establish the informational links, are not particularly associated with computation or the CTOM. In the philosophy of mind, information and computation remain substantially distinct; rather than viewing it as *information processing*, philosophers of mind mostly view computation in terms of the substantially distinct notion of *formal symbol manipulation*.

One of the primary briefs of AOS is to distinguish, and differentially assess, these and other competing construals of computation.

purposes. On the design side, numerous early programming languages took inspiration from the philosophical sides of logic and mathematics—including, most famously, Lisp,[9] one of the most preëminent programming languages ever designed, and for many years the *lingua franca* of artificial intelligence (AI).[10] Another celebrated example of direct architectural influence is Prolog,[11] which formed the basis for the Japanese "Fifth Generation Computer Systems" project during the 1980s.[12] Beyond their use as the basis of linguistic design itself, logical frameworks are widely used to analyse the so-called "semantics" of programming languages, including for example various forms and extensions of modal logic, which have been pressed into service to analyse concurrent processes and computational behaviour (e.g., in what is known as "dynamic logic"[13]). Recently, too, the pervasive influence of logic has been demonstrated in its use for the semantical model underlying most of the infrastructure underlying the internet and world-wide web, including OWL, common logic, semantical analyses of RDF, and proposals to orchestrate a "semantic web."[14]

---

9. For: 'LISt Processing language.' See «ref McCarthy», and Part B.

10. «Put in quotes from Alan Kay and others…where did I use those?»

11. 'Prolog' is short for "programming in logic—or, strictly, for *programmation en logique*, a name chosen by Philippe Roussel in France, who worked with Alain Colmerauer on the creation of Prolog in the early 1970s.

12. During a trip to Japan in the late 1980s, I asked a number of computer scientists and government officials involved with the Fifth Generation Programming Project why, if Japan felt it was appropriate to base major computational development on the formalization of a philosophical conception of human rationality, and if the project had nationalistic goals (which seemed evident), they had chosen to import, wholesale, a conception of rationality imported from and so clearly associated with the Anglo-American West, rather than to formalize historically Japanese or Buddhist conceptions of the mind. It is perhaps needless to say that the question was greeted with almost complete incomprehension.

13. «Refs»

14. One notable exception is Jean-Luc Girard's linear logic, which en-

So the connections between philosophy and computation are numerous. And yet, from the perspective of someone living at or in the intersection of the two fields, it is difficult, especially over time, to avoid feeling that the two fields collaborate in maintaining something of an abiding quietism as regards their interaction. Consider for example the title of a series of conferences that just recently celebrated its 25th anniversary: Computation *and* Philosophy.[15] In one sense, this epithet characterises my own work, of which some is collected in these volumes. I, too, have tried to bring the fields together—focusing on computing as primarily an empirical ground, and drawing on philosophy especially for analytic frameworks and methods. Yet it would be a mistake to characterize the relation between the disciplines manifested in the essays collected here in terms of a simple *conjunction*. More violently—curious phrasing for a committed pacifist, I know, but I believe the description is (or anyway should be) accurate—my goal has been to *subject* both fields to each other. In order to develop an industrial-strength understanding of meaningful mechanisms, I have increasingly found it important to bring the two fields together with sufficient ferocity as to leave neither unscathed.

I do not consider a philosophical analysis of computing to have more than scratched the surface of its subject matter, for example, unless it leads to genuinely novel, intellectually challenging computational architectures. In my book, that is, the merit of a philosophical analysis should be measured at least in part, not simply by its adequacy in providing an account of programming, but by the extent to which it *leads programmers to program differently*. We are infants in the computational realm, after all—mere alchemists, brewing up concoctions

joyed a period of popularity in theoretical computer science in the 1990s, and which looks perverse to anyone classically trained—for example in its *failure* to license the inference from p to $p \wedge p$.

15. Emphasis on the conjunction added; see below.

of marvelous but ad-hoc and (so I will argue) stunningly untheorized power. Sure enough, we can and should celebrate the astonishing power of synthetic computational processes. But if there is one moral I would like to demonstrate in these volumes, it is this: no matter how impressive our handiwork so far, especially from a practical point of view, the state of our theoretical understanding of computing is still remarkably inchoate—even confused. It is not that present-day practice is unworthy of study. On the contrary, I think it deserves far more philosophical attention and respect than it is normally accorded. But studying it should make at least this much evident: computing needs help as much as it needs observation.

Mutatis mutandis, no analysis of computation has cut deep enough, in my book, unless it unleashes reverberating philosophical insights, and in the process undermines, reconfigures—in some cases even wreaks havoc on—centuries-old staples of philosophical analysis. This is a strong claim, substantiation of which is one of the long-term goals of the papers assembled here. While I cannot hope to make good on it in this Introduction, it may still be worthwhile listing a few simple examples, to convey a flavour.

First, one form of philosophical consequence that arises from a serious examination of programs, processes, and computational practice has to do with such familiar (and paradigmatically binary) distinctions as those between and among *type* and *token*, *use* and *mention*, *intension* and *extension*, *syntax* and *semantics*, *analogue* and *digital*, and *abstract* and *concrete*.[16] No one of these dichotomous typologies, it turns out, can do justice to computational practice—not only to the complexity of practice, which is easy enough to demonstrate, if not ultimately all that interesting, but more significantly to the fundamental regularities on which that practice is based.

16. One could add to this list that between *mind* and *body*—but that is a more complex issue.

In one of the projects reported here, for example,[17] I attempted to "clean up" a spate of manifest use/mention confusions that plague the design of Lisp. In the design of the alternative I proposed, named 3Lisp, I aimed for, and believe that I achieved, what from a classic philosophical perspective would be considered a degree of theoretical elegance. When I was done, however, to my considerable retrospective surprise, I discovered that I had produced a language that, far from being unequivocally better than previous dialects, was in many ways *worse*. Ultimately, what is wrong with 3Lisp is as or even more interesting than what is right about it. I had achieved philosophical clarity—or at least clarity in terms of currently reigning philosophical norms—at a price of unusably fastidious baroqueness. Moreover, to up the ante, I also discovered, as soon as I tried to give theoretical voice to this recognition, that the reasons the results were intolerable did not find easy expression within accepted conceptual frameworks. Theory should be sobered.[18]

A second example. Philosophy is permeated with discussions of what, in general terms, can be called issues of *reduction*: how an analysis—or as I will say, "registration"—of a system at one level of abstraction or idealization (or in terms of one ana-

17. The design of 3Lisp; see «ref chapters».

18. Moreover, as explored in the final paper in Volume I ("The Correspondence Continuum," ch. 11), the project also not only highlighted but demonstrated the practical and theoretical infeasibility of maintaining another familiar and allied somewhat higher-order binarism, between: (i) a strict designational or referential hierarchy between signs (terms, descriptions, sentences etc.) and what they signify, of the sort regularly buttressed by strict use/mention distinctions; and (ii) a practice of promiscuous but theoretically invisible modeling, as for example typified in model theory and category theory, in which some entities are identified with others (typically: with entities considered to be isomorphic), without those intentional relations (modeling being a form or species of representation) being brought into theoretical view.

lytic or ontological or conceptual scheme) relates to another analysis or registration of that self-same system at a different level of abstraction or idealization (or in terms of a different analytic or ontological or conceptual scheme). For example, to consider the most familiar case: of how, assuming we are not substance dualists, are we to understand the relations between and among the following three levels of understanding the human mind: the psychological, the neurological, and the biochemical.

Philosophers have developed distinctive technical vocabulary to analyse and disentangle a variety of such issues, including such notions as *type-* and *token-reduction*, *local* and *global supervenience*, *modularity*, *mereology*,[19] *emergence*, and *holism*. For its part, computer science has developed theories and, more importantly, honed deep embodied practices and intuitions,[20] having to do with the very same or at least

19. Mereology because analyses of the part-whole structure are at least referenced to, and perhaps constitutive of, the notion of a system's being intelligible (to put it epistemically) or constructed (to put it ontologically) at a variety of different levels of "graining," abstraction, or idealization. In the general case, whole/part distinctions at one level need not correspond to whole/part distinctions at a subvening (philosophy) or implementing (computer science) level. It is not only obvious but even standard practice in computer science to develop implementations not only in which individual token objects at one level are not implemented by individual token objects at a lower level, but even (contrary to philosophy's conception of token reduction) where an individual *event* at a higher level need not be implemented by anything registered as an event at a lower level. How mereology and implementation relate, especially with reference to dynamics, is not a topic that, so far as I know, has been theorized in either field.

20. Computing being largely an embodied, engineering discipline, the "forms of knowing" pervasive in and constitutive of computational expertise often run deeper and are more subtle than has been captured in attempts to render them theoretically explicit. I comment in the text on the fact that the philosophy of computing deserves more philosophical attention than it has received to date, and that it might be considered a subfield within the philosophy of science (though, as discussed in

closely related issues. Unsurprisingly, however, computer science uses different vocabulary to talk about the issues, using such terms as: *implementation layers*; *black-box*, *grey-box*, and *transparent abstraction*; *protocol stacks*; *abstract data types*; *input/output specifications*; *application programming interfaces* (APIs); *modularity* (again), and a plethora of other notions. Yet to my knowledge no one has yet developed a careful analysis of the relations, similarities, differences, contrasts, etc., between these two traditions' attempts to wrestle with an enduring common problem.[21]

What matters about such a project, moreover, is that it would involve far more than mere translation. The situation is more reminiscent of the proverbial blind men and elephant. The integrative task would not merely be to articulate the insights of each in ways that the other could understand, but to incorporate their respective partial understandings into a deeper, more integrated account.

A third example, to show how a seemingly small and isolated technical point can have substantial philosophical reverberations. In philosophical discourse it is common, if not ubiquitous, to take seriously the distinction between things that are "possible in principle," even if they are hopelessly impracticable, versus things that are impossible, as it is said, *even in principle*. If a set of choices is finite, for example, then it is thought that "in principle" they could all be examined, whereas if the set is infinite, then exhaustive enumeration is taken to be out

AOS, its inherently intentional character may challenge this classification). But there are also ways in which it should be considered a subfield within the philosophy of *engineering*—another area that would be well served by increased philosophical attention, because its epistemic profile is different. At present, engineering and embodied practical knowledge are more directly theorized within science and technology studies (STS), as for example in the work of «refs».

21. For decades I have included this issue on an informal list—to use a little Pennsylvania Dutch—of "doctoral theses needing written."

of bound—and out of bounds in a way that is importantly distinct, philosophically very different it is assumed, from the way in which it is out of bounds to examine the set of all possible chess games (a finite number, estimated to be $\sim 10^{120}$).

This difference between the "merely impracticable" and the "impossible in principle" is taken to be of enormous epistemological and even ontological consequence—bearing, e.g., not only on the question of whether the world is deterministic, but also, if so, on whether, as some would have it, the truth of that fact impinges on the possibility of free will, and the like. The issue plays a role, too, on standard interpretations of the philosophical impact of the Gödel incompleteness results, Turing's proofs of the unsolvability of certain problems, etc.

Programmers, to put it bluntly, see things differently. The "absolute computability" results that captured philosophical attention in the 1920s and 1930s, cited above, have given way, not only in imaginative and practical importance, but also in theoretical weight, to the so-called *relative computability* or *complexity*[22] results, having to do with the difficulty of doing things—with just how hard it is, how hard it can be proved to be, in fact, to accomplish certain tasks. Some of these results are expressed in terms of what is called temporal *order*, in the sense of how much time a problem would take to solve (more generally: an algorithm would take to compute) as a function of the size of its input.[23] But even when the order is low or

22. See the discussion of complexity in fn. ■n, below.

23. Specifically, if the input is of size x, then the complexity order of an algorithm can be understood with reference to the character of the largest factor proportional to x in the equation stating the amount of time that the algorithm would take to complete. Thus if the amount of time is proportional to x itself (i.e., to the size of the input), the algorithm is called *linear*; if proportional to x^2, *quadratic*; if proportional to x^n for some integer n, *polynomial*; if proportional to e^x, *exponential*, etc. *(cont'd)*

As many readers know, the most famous open problem in the area of computational complexity theory is the as-yet unsolved issue of whether P=NP—i.e., of whether the class of deterministic polynomial algorithms

small—linear, or even finite—the amount of time can still be astronomical (chess being the standard example, which as mentioned is finite but still of a size transcending all practical possibility). The bottom line is that some, even many, operations are simply "too hard" to carry out, in anything remotely like the current universe.[24]

For programmers, that is, the difference between what is *practicable* and what is *impracticable* is more important than that between what is finite and what is infinite. To put it in the plainest possible language: the difference between small numbers and large numbers (where 'large' means '*very* large') is taken to be more important than between 'large' and infinite). The philosophical significance of this fact, moreover, is that it does not simply stem from programmers being an instrumental lot, focused on efficiency and pragmatics at the expense of theoretical or conceptual depth. The issue cuts deeper. Marinating in computational practice, in my experience, affects one's brain—leads one to a profound philosophical recognition that it is the distinction between what is practicable and what is impracticable, rather than that between what is finite and what is infinite, that is of theoretical, epistemological, and ontological significance.

Philosophical habits die hard. Some readers will resist this argument, claiming something along the following lines: that the difference between what can and cannot be done *in prin-*

(algorithms whose temporal order, given input x, is proportional to x^n for some integer n) is equal to the class of non-deterministic polynomial algorithms (very roughly, algorithms the correctness of whose answers can be checked in polynomial time).

24. Complexity calculations of this sort are easy to find on the web. Chess may be finite, but since the number of possible moves is estimated to be on the order of 10^{123}, the minimal unit of temporal variation approximately 10^{-35} seconds, the age of the universe about 10^{18} seconds, and the number of subatomic particles on the order of 10^{-80}, so the maximum number of individual atomic vibrations since the beginning of time is about $10^{35 \cdot 18 \cdot 80} = 10^{123}$.... «.....»

ciple is more profound than that between what is and what is not *practical*. But the very formulation of that reaction shows that it exactly misses the point. What computationalists have come to realize is that these *are* matters of principle. That is the point of all those theorems. *They are principles of practice*.

The issue is ultimately one of dynamics. Computer scientists, I believe it is fair to say, are among other things fundamentally interested in *time*. And although "computational time" is sometimes theorized abstractly, there is no ambiguity that it is real time—genuine temporality—that is at issue. Husserl is at point.[25] In philosophical discussions of computing, particularly in the context of the CTOM, it is famously said that computation is "medium independent," in the sense that it is deemed possible to implement any given computation on an unbounded number of different physical substrates. But no matter how arbitrary the details of the spatial and causal mapping,[26] the *dimensionality* of the mapping is in no sense arbitrary: that computational time be implemented as real time is absolute.[27]

Computational complexity theory is thus appropriately understood, philosophically, as a theory of *temporal principles*. So if it can be proved, in a principled way, that doing α would require billions of times more time than the universe has existed, or is predicted to exist, then to a computationally trained imagination α is *impossible in principle*—just as impossible as if it would take an infinite amount of time. To assume, as if it were an a prior God-given truth, that application of the

25. «ref Husserl, *Vorlesungen zur Phänomenologie des inneren Zeitbewusstseins* (1928), translated as *The Phenomenology of Internal Time Consciousness*, Indiana University (1964).»

26. And I am not convinced that they are *that* arbitrary. See AOS.

27. It is not possible to map the "temporal" dimension of a computation, no matter how abstractly viewed, onto a spatial dimension of the underlying substrate, and still call the result a computation—e.g., by laying out on paper the evolving state of a finite state controller. At best the result would be called a model or a representation.

predicate "impossible in principle" requires abstracting away from all issues of temporality comes across as the prejudice of someone who is, to be blunt, *temporally unprincipled*. And what kind of philosophical firm ground is that?

The issue does not stop there. A deep underlying theme of the theoretical transition that computer science has ushered in, from its origin in logic and the philosophy of mathematics in the early 20th century, through the next hundred years, has been a general shift from a focus on states and, as they have classically been called, "state transitions," towards a more direct focus on process, temporality, and dynamics. Think of how we used to theorize computations, thirty years ago, in terms of *memory*, *instructions*, *state transition functions*, etc., and contrast that to contemporary discussions, which are far more likely to be theorized in terms of *threads*, *forks*, *joins*, *agents*, *behaviour*, *processes*, etc. The analogy with classical physics is hard to miss. Surely the most important ontological insight in Newtonian dynamics was the recognition that, at least in the meso-scale physical world, what is regular and law-like, what repays deep theoretical attention, is *change*, with *state* being derivative. From a purely physical point of view state is ontologically no more than the accumulation of change over time through law-like transitions.[28] While it may not match our mundane world view or "natural ontological attitude," ontologically, *how things are* is derivative; *how things change*, primary.

Strikingly, philosophical theorizations of mind, epistemology, and logic have not made any such dynamical transition. In logic, we still give theoretical primacy to *truth*, viewing it as a normative predicate on *states*—what I elsewhere call a statical norm.[29] The task of temporalizing it is left to the poets and mystics—in their talk of "walking in truth," their characterisa-

28. Change that started out from those infamously diabolical boundary conditions.

29. 'Statical' because it is a *norm on states*—as opposed to *static*, which would mean that the norm itself would not change.

tion of "truth as a *way*." Then, given a secure footing in such a statical conception of truth, logic imposes an ontologically and explanatorily derivative norm on change—on transitions in proof theory, on moves taken by a theorem prover, on inference steps and inferences as a whole, etc.: that they be *truth-preserving*. Soundness and completeness are the same; they are derivatively defined, in terms of a prior conception of statical truth. Similarly in economics, when we talk about 'utility maximization,' and elsewhere throughout reigning intellectual regimes. In all these cases, the *dynamical norms*[30] are derivatively defined, in terms of what we take to be ontologically and explanatorily primary statical norms.

In my estimation, computer science has not yet developed very good theoretical machinery for talking about processes and dynamics[31]—a topic I return to in §6. But one would miss some of the most important facts about computing if one were not to recognize how much of computational analysis has nevertheless been shifted, torqued, and redeployed so as to move our focus from *states* onto *changes* and *behaviour*, from the *static* to the *dynamic*. The shift is not as manifest as it might otherwise be, because often the theoretical machinery used to effect this change is not superficially distinct; rather, classical apparatus has been morphed and torqued to target behaviour.

One case deserves specal attention here. Throughout these papers, I return again and again to the fact that, in computationalists' hands, the notion of 'semantics' has been repurposed to name the relation between programs and processes—i.e., the relation between static programs and the dynamic be-

30. That is: the norms on the dynamics, as opposed to dynamic norms, which would be norms that change. See fn. ■n.

31. «Ref dynamic systems theory, and its upsurge in popularity in cog sci. But that machinery is not only applicable only to continuous processes, but also, more seriously, to processes that can be assessed in terms of a numerically-valued measure property. Computational processes are not only in general disjoint, discontinuous, discrete, etc., but measure properties are nowhere in view.»

haviour that results from running or executing them—rather than the relation between the computation and the world or task domain that it would intuitively be taken to be about. The latter relation—the relation that philosophers would probably unquestioningly take to be *the* important semantic relation—remains by and large untheorized. It is because of this repurposing, I believe, rather than any deeper metaphysical commitments on computationalists' part, that intuitionistic type theory and constructive mathematics have grown so popular in computational theorizing. How to deal with the omissions and demerits of this practice I leave until later; for present purposes, what is telling about these developments is how they betray the theme currently in focus: an unadvertised and as-yet inadequately theorized but nevertheless hugely important shift towards bringing dynamics, process, and behaviour onto theoretically center stage.

Return to the topic that brought us here: how a serious examination of computing might not just exploit philosophical methods and insights, but affect philosophy itself. The three examples I have cited are relatively simple: (i) the inadequacy of numerous familiar but ultimately too-simple binarisms, (ii) a potential reconceptualization of our understanding of reduction and implementation, and (iii) a shift towards a more dynamic conception of principle and a granting of ontological primacy to dynamics and process. Imagine what it would be to bring these insights and adjustments into our understanding of mind, just to take one obvious case. Suppose computing, once philosophically understood, could lead us to develop analytic tools for understanding the ways in which the *saying* affects the *said*. Suppose we constructed tools and techniques for understanding implementation, particularly implementation of process, and could bring theoretical equipment to bear on our understanding of the relation between consciousness, mind, the personal and the subpersonal, and the relation of

all to the neuroscientific. Suppose we could develop a genuinely dynamical conception of norms, rather than defining the norms on thinking (which after all is a process) derivatively in terms of a statical conception of truth. And so on. This is what I imagine would come from a serious philosophical investigation of computing. It would affect our understanding of *what mind is*. The computational theory of mind would no longer simply refer to *the mind as computational*, understood to mean something like this: that cognition, roughly as presently understood, arises in or supervenes on a computational substrate. Rather, and surely much more interestingly, it would suggest something more like a theory *of the computational mind*.

Nor would mind, and the analytic philosophy of psychology, be the only philosophical topics affected. A very quick example. In a recent dissertation,[32] Kevin Eldred conducted a phenomenological analysis of the mode of being enjoyed by avatars in the massively multi-player online game World of Warcraft,® arguing: not only (i) that contemporary analyses that view avatars as tools, prostheses, virtual bodies, fictional agents, and a host of other familiar ontic possibilities largely miss the point; but also (ii) that sustained serious play in such communities is a genuine (not inauthentic) form of world-disclosure, worthy of sustained phenomenological analysis; and also—and this is where it gets interesting—(iii) that such analysis not only illuminates the realms of online gaming, but puts pressure back onto our understanding of phenomenology itself. In particular, he raises the probing question of whether, even if "death" in such online games is virtual rather than real, *authentic Being unto virtual death* may not have implications for Dasein different than, and perhaps in some ways more illuminating and instructive, than the classically theorized *inauthentic Being towards real death*.

It is not my purpose here to weigh in on this or any of the other possibilities adduced above. The point is more general.

32. «ref»

Computing is not merely a medium, or a substrate, or a market opportunity for philosophical reflection. The reconfiguration of the age of mechanism and the materialization of the semantic and normative constitutive of computational practice are topics that should and will reverberate throughout philosophy for centuries to come.

What about the other direction—the impact of philosophy on computing? Again, the CTOM is a useful foil in terms of which to ask the question. On a schematic view of the thesis as something like the equation "mind ≈ computing," or "minds ⊂ computers," one might imagine the impact to be symmetrical, affecting our understanding of computing as much as our understanding of mind. To my knowledge, however, that direction of implication is rarely explored. In fact the thesis underwriting the CTOM is generally understood along something like the following lines, I would hazard, with computation taken as already theorized:

1. **Theory-laden:** Given that computing is as we think it is—formal symbol manipulation, paradigmatically—the mind is *like that* (is implemented as that, is to be understood in terms of that, or whatever).

In my teaching, and for many years in my own research, I have rejected this theory-laden reading in favour of something more empirical and ostensive:

2. **Ostensive:** The mind is like computing (is implemented as a computation, is to be understood in terms of computing, etc.), *whatever it is that computing is like*.

The ostensive formulation makes the CTOM doubly empirical, a thesis about computers as well as about minds, by opening the question of what computing is—the subject matter of AOS, and, in my view, a stunningly open and as-yet unresolved issue. But neither formulation brings forward a third issue, di-

rectly pertinent to the question of the engagement between computer science and philosophy:

3. **Reflexive:** What would computing be like, if computers were like minds?

In Turing's original article, of course, the very notion of 'computing' is characterised with reference to human activity (the activity of "computers," who in his day were people—mostly female). In recent decades, I believe it is fair to say, the human as a source of inspiration for the nature of computing does not receive much attention, even if human activities provide an inspiration for certain kinds of computational architecture.

Yet this third reading has also pervaded my own work, and figures substantially in the papers in this volume—a reading, that is, in which the impact flows from mind to computing as well as from computing to mind. As already intimated, the issue is especially pertinent to the issue of semantics. As explained in numerous ways throughout the following pages, my take on semantics parts company with prevailing practice in computer science; the present point is that it ties much more directly into semantics as it is understood to be a property of human language and thought. Both in this volume and in the next,[33] I not only take inspiration from the human case, but in fact take the nature of representation and semantics and the like to be paradigmatically as exemplified by people, and then apply those understandings to the computational case—assuming that, if computers represent, they must do at least some kind of justice to the nature of representation as exemplified by human activity. Similarly, my arguments about the non-causal (non-effective) character of semantical properties such as that of "being referred to," and my insistence on theorizing what I call the "declarative import" of programming languages, clearly derive from an understanding of the human case.

33. See in particular "Representation and Registration," in Vol. II.

And so on, for a myriad other issues. And to return to the issue with which this discussion started, having to do with the character of engagement computer science and philosophy, I am not saying that rapprochement would be easy—or even suggesting that 'rapprochement' is the right word.[34] The point is merely this: if we are to get to any such future, the encounter of the fields will need to be much more forceful than it has been to date.

2 Etiology

The essays collected here do not collectively present a stable and rigorous theory of anything. As regards the foundations of computer science and philosophy, they do not even achieve fission, let alone fusion. Rather, they are presented as something of an historical record, documenting one person's struggle, over several decades, to investigate a variety of issues on which the two fields at a minimum overlap, on which they have developed contrapuntal intimacies, and where possibilities of deeper engagement are evident.

The papers in Volume 1 were primarily written between 1982 and 2002. I might say that they start out investigating a variety of problems having to do with the ontology and semantics of computing—though it should be admitted that that is a relatively content-free claim. By *ontology* I basically mean what computation is, and within the realm of *semantics* include pretty much everything else: how it relates to the world. Nevertheless, since the language in terms of which the discourse is conducted includes both ontological and semantical terminology, that is the frame from which they are best initially approached. A subset of the papers are devoted to an analysis of what I called *reflection*—a computational architecture in which systems are constituted with explicit reference to ingredient self-descriptive theoretical models, in such a way as

34. See the sidebar on "Computer science as an empirical inquiry." «???»

to provide them with the capability of both "reasoning about"[35] and modifying their own behaviour. As becomes evident in the ensuing discussed, moreover, because of the self-descriptive aspect,[36] little about computing escapes the compass of those reflective deliberations.

Accompanying each paper, I have included a short introduction, in an attempt to provide some explanatory context. These introductions—or "covers," as I call them—are far from neutral; they unapologetically reflect a particular (my own) 2012 perspective. As well as briefly indicating my current view of each paper's achievements, negative as well as positive, I say a few words about how I would approach the issue were I to tackle it today.

The papers in Volume II are by and large more recent, most having been written in the first decade of this century. In contrast to those of Volume I, most are previously unpublished. Though the first several focus on computation, and others make use of computing as an example, the emphasis gradually shifts, so as to become more bluntly metaphysical. Still, the distinction between Volumes I and II is more historical than thematic. Some of the papers in the first Volume—especially including "Varieties of Self-Reference" and "The Correspondence Continuum," but also "The Semantics of Clocks"—tackle subjects, such as self-knowledge, correspon-

35. Quotes because the system is not a reasoning system; see the introduction to Section ■n.

36. See "Varieties of Self-Reference," ch. 6 of Volume I, for an analysis of why reflective self-description goes beyond the more philosophically familiar topic of self-reference, as studied for example in logic («ref Barwise *The Liar*»). The model of computational reflection is more related to so-called reflection principles in logic, though the complexity of the issue in the computational case, and I believe the interest as well, is considerably higher, because of the critical inclusion of dynamics. From an epistemic point of view, one might say that reflection has more to do with an integrated combination of practical reasoning and self-knowledge, rather than merely with self-reference.

dence, and the measurement of time, that, though critically applicable to computational contexts, are in no way inherently computational. And while some topics considered in Volume II, such as that of non-conceptual content, may not look computational on the surface, it will be evident on anything more than a cursory reading that my analysis is deeply affected by computational examples and experience.

In part to make sense of this range of topics, but more specifically to explicate the particular ways in which I approach them, I want to make four general framing comments here. As a set, the papers were written from within an intellectual context that, though not explicitly addressed in any of them, it will help to understand, not only in order to see how the individual pieces fit together, but also to make evident how they can be understood—especially in retrospect—to present a coherently developing line of thought.

3 Computing

I have been interested in the fundamental nature of computing for more than four decades.[37] By as early as the mid-1970s, I had come to believe that all existing theories of computing are inadequate, and for very deep reasons: conceptually confused, empirically awry, unable to do justice to what matters about computational practice, and inappropriately fundamentalist in a variety of ways—e.g., the mathematical theory of computing's presupposition, rather than, at least in my view, anything like a compelling demonstration that computation should be treated as an abstract phenomenon, rather than as something that is in any essential way concrete. At first, as spelled out in more detail in the introduction to Section B, my concerns were primarily focused on semantics—though over time, as already mentioned, and as explained in more detail in §6, below, my sense of the difficulties shifted to include more challenging considerations of ontology and metaphysics.

37. Specifically, since November of 1968; see «ref 03».

Overall, my efforts to take the measure of computing have been conducted into two broad parallel, complementary paths. Just one of the paths is presented in the papers collected here. At least in comparison to the other, this first path is relatively more focused and technical, and investigates specific issues—*reflection*, *correctness*, *semantics*, *programming languages*, *computational semantics*, and so on—in ways that not only aim to shed light on those topics in particular, but to do so in such a way as to open up and deepen our understanding of computation at a more fundamental level. The second path, underlying the first, has involved conducting a full-scale, systematic investigation of the philosophical foundations of computing, in an effort to develop more adequate theoretical understanding, in terms of which not only the particular issues that I have chosen to concentrate on in the first path, but computational phenomena in general, can be more adequately framed.

Results from the second path are not included here. As has been promised for more decades that I care to number, the intent is to publish the efforts in that direction separately, under the general title *The Age of Significance* (AOS).[38] That work is expected to comprise a series of volumes that wrestle with the surprisingly wide variety of notions, issues, themes, assumptions, and claims on which our current understanding rests. Metaphorically, the AOS volumes can be viewed as reports on an effort undertaken in the basement (even sub-basement), reworking the most basic assumptions and theoretical presumptions—in an attempt to work towards more adequate footings.[39] The projects in the first path, discussed here, have

38. They are being made available online as well as on paper; see http://www.ageofsignificance.org

39. One of the aims of that project is to reinstate and revive fundamental and foundational concerns, notwithstanding both modernist and postmodernist allergies to metaphysical foundations. As already suggested in *ch. 2 of O3*, I believe it is possible to take on the issue of founding an account (and a vision of life) in the world, without running afoul of the well-rehearsed and compelling flaws of previous "metaphysical"

by and large been conducted at higher storeys. They address a variety of phenomena not themselves necessarily or obviously computational, though in ways that honour computational understanding and practice.

In order to show how these higher-level papers should fit into that longer-term, more systematic analysis, I have included, as the first paper in this collection, a brief (and dense) summary of the first phases of that underlying ("second path") investigation.[40] For purposes of this Introduction, what matters about that paper is its main conclusion: that what starts out as a search for a more adequate theory of computation ultimately fails. The reason for the failure is not, I claim, that because while a better theory of computing may someday be brought forward by someone else, I myself have succeeded neither in finding nor in developing such a thing. More seriously, I claim that a more adequate theory will never be possible. We will never have an adequate theory of computing for the simple reason that, contrary to what as far as I can tell is almost unanimously otherwise believed,[41] *computation is not a subject matter*—not, as philosophers might say, a "natural kind," not the sort of thing of which a deep, penetrating, useful theory will ever be developed. Rather than being like tigers, or neutron stars, or possibly even democratic principles, computers, in my view, are more like *cars*—objects of inestimable practical value, but as regards theoretical reconstruction more a madcap bricolage of parts and ideas and processes that reflect our best emerging sense of how to build powerful meaningful machines, rather than anything of specific theoretical or intellectual interest.

One of the briefs of the AOS series, as well as providing sup-

and foundational approaches. Not incidentally, some of the intuitions as to how this can go have developed out of my experience in building computational systems.

40. "The Foundations of Computing," section A.

41. Philip Agre is a possible exception. «discuss»'

porting arguments and evidence for this "not a subject matter" claim, is to argue, first appearances notwithstanding, that this conclusion should be viewed as exhilaratingly positive, rather than depressingly negative—that it should warm the heart of even the most unregenerate computational triumphalist. What it means is that computation should not be understood as a *specific*, which is to say *restricted*, kind of thing[42]—digital machine, formal symbol manipulator, information processor, or anything else—but rather as an *unrestricted laboratory* in which to explore issues of meaning and mechanism.

The implication of that conclusion for the issues explored in these papers—reflection, self-reference, general correspondence, semantics, time, etc.—is both straightforward and strong. Appearances notwithstanding, to put it as simply as possible, it means that the insights and architectures and challenges and solutions developed in the computational arena are not specific or particular to the computational case, since there is no "computational case" to be specific to, but are instead insights, architectures, challenges and solutions about reflection, self-reference, correspondence, semantics, etc., in general.

Needless to say, this conclusion considerably ups the ante on the point made at the outset: that a philosophically trenchant account of computing should have an impact on philosophy—not just be of philosophical interest or importance, but actually *affect philosophical analysis itself*. If I am right that computation is not special—not a natural kind, not a subject matter worthy of study per se, not a subject matter of which a satisfying or penetrating theory will ever be developed—then, as just suggested, studies of any phenomena in the computational realm, be they self-reference, reflection, semantics, etc., or any other topic that I myself have not pursued, are no less than studies of self-reference, reflection, semantics, etc. *tout court*. Fusion is thus a better metaphor than fission (and any-

42. I.e., 'specific' in the literal sense of constituting a species, subordinate to some higher order genus.

way the released energy is higher). Philosophical insight and computational understanding do not just bear on each other; they should be *merged*—forged into a single account that incorporates what is best in both, rather than remaining at a safe and self-preserving distance. Hence the sentiment hinted at above: projects in philosophy and computation, philosophy of computation, computational philosophy, and the like, should not be applauded. They should be replaced.

Two immediate comments, to deflect misunderstanding. First, it is not my intent to exaggerate present achievements. I make no claim that contemporary computer science has developed insights relevant to—or, rather, part of—moral theory, or political philosophy, or aesthetics. At least not yet. But if, as suggested above, and as argued in AOS, computational practice is indeed best understood as an unrestricted laboratory for the exploration of meaningful material systems, then we are virtually destined to encroach on such territory in due course. The issue is not simply that we will someday develop technical systems about which ethical issues arise. That has been true for a long time already (what is the appropriate ethical stance to take towards failures in mandated medical systems—or towards avatars, or the people "behind" them, that commit crimes in massive social virtual worlds?[43]). Rather, only blind-

43. Consider another example. Beginning in the 1970s and 1980s, automobile manufacturers introduced computerized braking systems (known as 'ABS,' from the German *Antiblockiersystem*), which can detect skidding and control the braking force of each wheel separately, and which for that and other reasons are better than any person can be in slowing a car down rapidly. Their wide-spread introduction demonstrably saved human lives (though, ironically, there is also some evidence that people drive just that much faster to maintain the same death rate as before. «ref») The curious ethical fact is that, whereas the number of deaths per, say, 100,000 miles of driving is reduced, the systems will inevitably occasionally fail, and although they are designed to be as failsafe as possible, and to devolve to regular braking if they do fail, some failure and even human death due to malfunction is inevitable. Moreover, the

ness to the magnitude of impending domestication of the physical substrate of human life by computing, genetic engineering, nanotechnology, and their combination—only rank human chauvinism—would deny that that we will eventually, and perhaps sooner than society is ready for, build systems that are *themselves loci of ethical responsibility*, such as machines that can make their own aesthetic judgments, machines that should be punished, or at least censured, when they do things that are wrong, machines that it will be unethical to unplug or bring to a halt, machines, to put it in philosophical terminology, with autonomous and original intentionality (or at least as partially autonomous and original as our own).

Second, and relatedly, some people will insist that the computational realm will at best give us *empirical* evidence for this, *practical* experience of that, and so on, without impinging on transcendental, normative, or other "genuinely philosophical" considerations. How they know this, especially a priori, escapes me, I confess—that is, how they know, a priori, that computational engagement is somehow restricted to the realm of the empirical. As much could be argued for our children, or our texts. "But computers are just machines!," someone will argue. Really? Have they done a controlled test?

Eschewing dualism is strong stuff. And don't forget the

---

small number of people who are hurt or die due to their malfunction will almost certainly be different from the people whose lives are saved in virtue of their introduction.

If a limousine driver "fails," in such a way as to cause a fatal accident, we are liable to blame the driver. If an ABS system fails, we may *say* that we "blame the braking system," but the system is not subjected to ethical censure. "It was just a breakdown of a physical system," we might argue. As neuroscience proceeds, we are destined not only to understand the brain and its role in choreographing behaviour better, but to have a detailed understanding of the neurological substrates of unethical behaviour. So the driver, too, will be "subject of a breakdown of a physical system." … *[[is this worthwhile? … perhaps not]]*

Maharal of Prague.[44] Yes, we are *in*, *of*, and *about* the world—but so too are our creations. Any proposed argument that the *way* we humans are in the world necessarily differs from the way that computers are (and will be) in the world requires an account of what it is to be a computer. And an account of what it is to be a computer is exactly that which, I argue, we do not, and will not, have. Dasein is being unto death? And computers do not die? Well, someday we may not die, either. Dasein will have to catch up.

At least as this is being written, many will find talk of such developments to be the stuff of science fiction—or, worse, culpably imperialist, or pathetically shallow. I trust it is clear that, as regards the depth of the human condition, the profundity of philosophical reflection, and the gravity of the stakes, my sympathies lie with such critics. It is on the innocence and containability of the technological that I differ. It can hardly be recommended that Asimov's laws of robotics be considered in graduate courses on value theory. But given the relentless pace of computational development, it is inevitable that we will soon start to build, if we are not building already, systems and devices and processes to which substantial, not superficial, normative and ethical considerations apply. And think about biology. It was not that long ago that it would have seemed reductively profane to suggest that altruism, or fidelity, or proclivities for truth-telling could be illuminated by biological analysis, or, for that matter, hosted in "merely biological" creatures. To dream that computing will stay safely impounded in a machinic stockade will soon seem equally quaint.

4 Terminology

The status of computing as a laboratory rather than subject

44. Rabbi Judah Loew ben Bezalel, the late 16th century chief rabbi of Prague, who (at least since the story was popularized and likely invented in 19th c. German literature) is alleged to have created a Golem.

matter is the first issue to keep in mind while reading these papers. It leads to a second, which bedevils the papers collected here: that of theoretical vocabulary. As is again explored in more detail in AOS, where some of the historical threads are disentangled, the various fields and disciplines that study the general category of what, as already indicated, I in general characterize as **meaningful mechanisms**—including at least computer science, philosophy, psychology, linguistics, artificial intelligence, and cognitive science—are distinguished by using a large number of common terms and concepts, but alas in multiplicitous ways, with diverse meanings, connotations, and implications.

The problem is well-known, and cuts deep. It is evident even to a beginning student that such terms as *concept*, *function*, *symbol*, *meaning*, *reference*, *interpretation*, *identifier*, *procedure*, *process*, *argument*, *value*, *object*, *number*, *ontology*, and a spate of others are differentially understood by theorists of different stripes. So long as literatures remain disciplinarily confined, confusion can be locally minimized, even if divides between and among disciplines are thereby deepened. But the terminological overlap raises problems for interdisciplinary and multimethodological conversations in general,[45] and is certainly an issue in these volumes. Virtually all of the technical terms I employ will be differently understood by different readers, depending on their intellectual and disciplinary background.[46] Moreover, the problem is not just limited to large-scale notions, as an example will illustrate. Though it seems surprising in retrospect, it was only after working closely for two years with the logician Jon Barwise, including on some relatively technical construction projects, that we discovered that the seemingly confined, technical notion of *binding a variable* was differently understood in logic and computing.[47] Moreover, as

45. «ref "The End of Meth"»
46. «quote "ontology is hot" ad in a London paper»
47. «explain: 'bound by' and 'bound to'; and Δs on "whether it is bound".

detailed in the introduction to section C, and presaged in the next section of this Introduction, even when the overt meanings of a term are disambiguated between disciplinary contexts, issues of assumption, connotation, methodological bias, normative standard, etc., remain lurking in the background, waiting to sew confusion and misunderstanding.

In my teaching, as well as mentioning this terminological problem explicitly, I sometimes distribute an informal crib sheet, consisting of a matrix with a dozen or more terms on rows, crossed with half a dozen fields in columns, with each entry containing a brief synopsis of how that term is used in that field. But just being aware of the problem, and knowing a single sentence about the differences, is not enough to ensure clear communication, since (as in the case of the crib sheet) the problem recurses when one tries to explain what a word means, or how it is being used. When linguistic or terminological confusion engenders misunderstanding and cultural clash, that is, semantic ascent is not a guaranteed route to resolution.[48] One does not thereby gain access to a shared "metalingua franca," an unambiguous common ground, in terms of which to explicate differences—a stable meta-level where both parties can adopt a unitary shared perspective in terms of which to survey the multiplicitous distinctions below.[49] In my experience, semantic *descent* works better: substantive engagement with examples. Not just adversion to examples, either, though that may be of some help, but examples need to be pointed to with language, and so can misfire. Shared concrete practical engagement with examples is more reliable—invitations to the lab, collaboration on mutual normatively-laden

And then go on, in a second ¶, to talk about the Lisp, Language, and Literature course at Stanford, and my suggestion that CSLI start a technical term library.»

48. «Ref BHS "The Microdynamics of Incommensurability."»

49. «cite BHS»

projects, working together in the trenches, and over dinner.[50]

In the essays included here, I have tried to be as intuitive and clear about the use of terms as I can, but I have no illusion that difficulties will not arise. They certainly arose, in computer science, in response to the 3Lisp papers I wrote in the 1980s, some of which are included here in Part B. For the record, it is perhaps worth saying that the primary disciplinary practices I have written out from are computer science, logic, philosophy, artificial intelligence, and cognitive science. Psychology and linguistics have been less central in my thinking, and my vocabulary is unlikely to reflect their biases. And no one has ever confused me with, or taken me for, a social or cultural theorist of any stripe.

One task I have undertaken, in the introduction to each section and individual paper, is to add explanatory footnotes in some of the places where I am aware that confusion may arise from perspectival differences in vocabulary. I have also indicated, in those introductions, some of the places within the papers themselves where terminological missteps are most likely. But as always, these comments reflect just one person's (in fact: the same person's) view. Readers may want to make their own annotations, as they make their way through.

5 Mechanism

A third issue permeating these papers has to do with the notion of *mechanism*, and with the role of mechanistic explanation. Two considerations bring mechanical or mechanistic issues to the fore in computational investigations generally, and especially those of the sort presented here.

50. The issue is hugely complicated by different metaphysical views—whether readers are Platonists, formalists, intuitionists, nominalists, constructivists, etc. The problem is that whereas those terms are common knowledge in philosophy, one cannot necessarily advert to them in discussions in artificial intelligence and computer science—though adherents of these and other metaphysical positions are as distributively exemplified as in any other arena.

5a Effective computability

To start with, it is universally agreed that computation has something to do with mechanism—with *what can be done*, with *what is effective*, with *what behaviour* it is possible to construct a device to exhibit. That agreement is only prelude to a raft of questions: (i) whether the relevant notion of a mechanism should be treated abstractly or concretely; (ii) how time should be handled—ignored, treated topologically or metrically, or something else; (iii) whether computing is restricted to *formal* mechanisms, or to *digital* mechanisms, or to both—or whether it includes, at an appropriately abstract level of abstraction, mechanisms of any stripe; (iv) whether it is adequate, in giving an analysis of a computational process or device, merely to characterise a mechanism's input-output behaviour, or whether one should or must refer to internal (causal?) ingredients—and even then, whether those ingredients should, recursively, be characterised in input-output behavioural fashion—i.e., to illustrate a terminological issue of the sort discussed in the previous section, whether those ingredients should be treated in ways that in philosophy of mind, but definitely not in computer science, would be called *functional*, rather than in terms of material constitution; and (v) whether mechanisms should be functionally or constitutively characterised, however that difference is made out. And so on and so forth. And underlying them all is the background issue of what it even is to be mechanical in the first place—a question that takes one back at least to Newton's discussions, in the *Scholium*, of what he took to be the non-mechanical character of gravity.

Yet in spite of this cornucopia of views, it remains a durable truth that, somehow or other, in ways that it is incumbent on a theory both to demonstrate and to explain, concerns with the mechanical or the mechanistic are invariably at the center of the computational question. Think of the most highly regarded mathematical theory in all of computer science: it is called a theory of *effective computability*. As already suggested,

I have troubles with the second of those two words, but none whatsoever with the first.

It is common, in fact, for computational sophisticates to think of computer science as something like the epitome of our understanding of mechanism—the apex of a tradition reaching back to at least the beginning of the Scientific Revolution. Challenging that characterisation is the additional fact that computing can also be taken, as we have already seen, to incorporate large swaths of logic, reaching all the way back to Aristotle, which brings other issues into play—especially considerations of meaning, reason, rationality, and mathematics. But if we just maintain focus on the effective, which anyway is likely the preference of most computer scientists, many would take the development of computing to represent something like a triumphant consolidation of the mechanist philosophy.

5b Blanket Mechanism

A second consideration brings the topic of mechanism to the fore in these pages, wider than anything recognized as specifically computational, and thus of special interest here: the methodological (rather than substantive) presumption that *good* scientific explanation must be *causal* explanation. I will leave general discussion of this methodological commitment to another occasion;[51] for present purposes it is enough to say that, especially among scientists, but in many quarters of philosophy as well, a commitment to causal or "mechanistic" explanation is remarkably widespread. Especially among contemporary students, that "explanation" means *causal explanation* seems to be assumed virtually a priori.

The juxtaposition of the these two facts—the substantive one, that issues of effectiveness and mechanism are and should take a central or even defining role in the computational realm, and the methodological one, currently espoused by so many people, that causal or mechanistic explanation is an unquali-

51. «ref AOS»

fied good—leads to a phenomenon that for discussion I will call **blanket mechanism**: an often implicit but nevertheless resolute tendency to assume, in discussions of computing, that *the entire theoretical discourse should be restricted to what is mechanical or effective*.[52] It is almost as if, in the face of these paired considerations, the embrace of mechanism transcends both ontological thesis and methodological commitment to become a wholesale and engulfing governing metaphysical norm.

Blanket mechanism is nothing I support. Perhaps because I have never embraced it, I had no idea, when the papers in these volumes were written, how widespread blanket mechanism has become, especially among computer scientists. One reason I did not appreciate the fact has to do with the issue discussed in the previous section: that of vocabulary. In spite of the centrality of the effective and mechanical in the contemporary computational imaginary, it is striking, as already mentioned, that the technical terminology of computer science was overwhelmingly taken from logic—as evident in such still-familiar notions as *symbol*, *reference*, *identifier*, *semantics*, (programming) *language*, *interpretation*, etc. And in those logical traditions, especially in their historical guise, those notions lie squarely on the meaning side of the meaning/mechanism dialectic.

Because I was interested in representation and semantics, and because from both historical and philosophical viewpoints these mainstays of computational jargon are semantical or intentional terms, I innocently assumed that, in computational discourse, they retained their semantical meaning.[53] But I was wrong. What has happened over the last fifty years, as I

52. I am intentionally not weighing in on the obvious question of whether I take blanket mechanism to be a metaphysical or a methodological commitment. … «expand»

53. «Footnote on 'semantic' vs. 'semantical'. "Semantic meaning" would be redundant; but 'semantical meaning' is not.»

have already suggested, mention in passing at several points in these pages, and explain in more detail in AOS, is that, under the aegis of a tacit, unquestioned submission to blanket mechanism, all those words have been re-interpreted or given new meanings, in ways that convert them into names for mechanical or mechanically explicable phenomena and relations.[54]

This is one of those statements that is likely to mean different things to different readers, and to provoke different reactions.[55] And that is the point. One of the most sobering realizations I have come to, over the last several decades, is how profoundly differences in interpretation and perspective on these issues have blocked communication and impeding understanding. So the issues are worth spelling out in a bit of detail. With luck, a bit of effort clarifying the issues here will aid in understanding the papers to follow.

5c Logic

Classically, it was never assumed that what is real or exists is restricted to what is mechanical or effective (effectively reachable, effectively constructible, etc.). For a particularly relevant case, consider Turing's original proof, in his original 1937–8 paper, of the limits of what can be effectively computed. In a nutshell, the proof has the following structure. Let $\mathcal{F}$ be the set of all mathematical functions, and $\mathcal{E}$ the set of functions that can be effectively computed by a machine. Then what Turing proved is that the set of computable functions is *strictly smaller* than the set of functions: $\mathcal{E} \subset \mathcal{F}$. That is: to say that there are

54. As some readers will immediately suspect, advertence to mechanical explanation to understand relationality is particularly problematic. See AOS.

55. "It is just false to say that interpretation is mechanical!," a philosopher might exclaim. "Interpretation is the semantical phenomenon *par excellence*." "Of course semantics is causal!" a programmer might say. "Don't you remember Newell: 'access to the object…is the essence of designation'. Don't you realize that we are *scientists*!" «Newell and Simon 1975, p. 116»

function than cannot be computed is to say that there are elements of $\mathcal{F}$ that are not elements of $\mathcal{E}$.

For this claim and proof to make sense, the set $\mathcal{F}$ must be taken to be perfectly and unproblematically *real*. In fact the very idea that some functions cannot be computed requires that those non-computable functions be metaphysically first-class. As first-class metaphysical entities, they exemplify numerous properties, one of which is the (again, perfectly real) property of *not being able to be computed by mechanical means*. Thus consider Turing's negative resolution of the halting problem—i.e., his proof that it is impossible to construct an effective procedure to decide whether an arbitrary machine $\mathcal{M}_i$ will halt on an arbitrary input I_{ij}. From the point of view of the proof, there is nothing metaphysically problematic about whether $\mathcal{M}_i$ will halt on I_{ij}. Either it will or it will not. There is a determinate fact of the matter, a fact that is metaphysically secure—a fact that would be manifest to an omniscient God. All that Turing proved is that, as creatures made of clay, we cannot have a general algorithm for *determining what fact that fact is*.[56]

Thirty years ago, I would have taken these points to be so elementary as to not have been worth including in a text such as this (if not in fact outright insulting). What I have been struck by, in the interim, is how pervasive the idea has become, particularly among students in cognitive science as well as computer science, that if something is not computable it is thereby somehow *metaphysically deficient*—may not exist, may not be a secure denizen of the pantheon, something like that. I am not sure, if one were sympathetic to such a view, how—or

56. To "determine what fact a fact is" is more often described as "deciding the fact—i.e., deciding whether $\mathcal{M}_i$ will halt on I_{ij}. To "decide" a fact, or to "decide whether a fact is true or false" has to do with representing it in some way—typically, it in terms of such canonical names as *True* or *False* ($\mathcal{T}$ or $\mathcal{F}$, or $T or $F in 2/3Lisp, etc.). See AOS—especially Volume III.

even whether—one could express the fundamental computability limits. But however that story might go, the success of computation has apparently led to, or anyway this has been my experience teaching over the last several decades, an upsurge in the number of strong *metaphysical* (not just methodological) constructivists, nominalists, formalists, and other stripes of mechanist appeal.[57]

Needless to say, not being a blanket mechanist, I do not believe that theoretical discourse must or even should restrict itself to the mechanical or effective, should limit itself to painting pictures on the mechanical wall of the cave. Nor, at least in the usual meanings of the term, am I a strong metaphysical constructivist ("social constructivism" is a different thing). These facts are crucial to understand, in reading the ensuing papers.

Consider in particular the semantical framework I developed for 2Lisp and 3Lisp, based in large part on my understanding of classical logic, and discussed in the papers in Part B. In accord with a classical approach, though the framing is my own, I viewed (and continue to view) logic as consisting fundamentally of a half dozen ingredients:

1. A syntactic domain $\mathcal{S}$, of expressions;
2. A semantic domain $\mathcal{D}$;
3. An interpretation function $\mathcal{I}$ mapping $\mathcal{S}$ into $\mathcal{D}$;
4. Two relations on $\mathcal{S}$:
   a. Derivability ($\vdash$): syntactically or formally defined; and
   b. Entailment ($\models$): defined in terms of $\mathcal{I}$ and a semantic consequence relation on $\mathcal{D}$; and

57. Note that this has nothing to do with *social constructivists*: those who believe that categories, or types, or objects themselves, are in whole or in part the result of human projects or understandings. One can perfectly well be—in some ways I myself am—a social constructivist without endorsing the idea that, for something to exist, it must be computable.

5. A *norm*, placing conditions on how the whole system is tied together.

The domains, relations and norm are depicted in figure 1; a bit more detail is provided in the sidebar on pages xliv–xlv. Again, at this schematic level, I take this structure to be elementary. I also assumed, when I wrote the papers included in these volumes, that at least in overall character it was something on which everyone would agree, and thus did not need introduction or explanation.[58]

As soon became evident, however, that was an illegitimate assumption, at least in computer science.[59] To see why, we

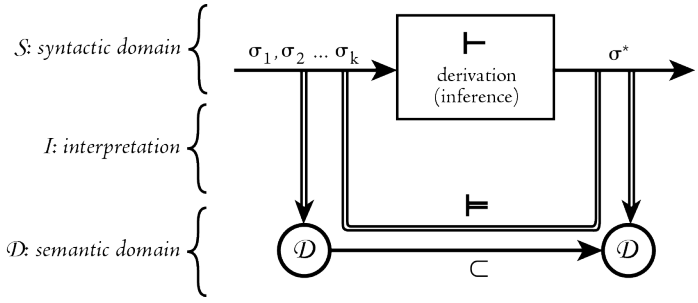

Figure 1 — Logic, schematically

need to answer the following critical question: which of the domains $\mathcal{S}$ and $\mathcal{D}$, and *which of the relations* $\vdash$, *I, and* $\models$, are assumed to be subject to mechanical constraint?

To make the question precise, note that "being mechanical" or "being effective" is a higher-order property—i.e., is not something defined on objects or properties or relations in extension, but, like being *causal* (another higher-order property, which effectiveness certainly resembles), defined over proper-

58. I am ignoring, for these purpose, the issue of whether one models one or more of these domains or relations mathematically; see "The Correspondence Continuum," Chapter ■n.

59. See the introduction to Part B.

Logic

As summarized in the text, I understand a system of formal logic to consist of two domains, three relations, and a norm:

1. Domains:
   a. A *syntactic domain* $\mathcal{S}$,[†] consisting of a set of expressions or formulae σ_i, written or constructed in a formal, recursively-specified, compositional language, where *formal*, as well as meaning precise, unambiguous, determinate, and a number of other things, is taken to imply that the syntactic properties of the expressions are defined without regards to the semantic interpretation (I, below); and
   b. A *semantic domain* $\mathcal{D}$, whose structure is subject to no a priori constraints whatsoever, though for the logic to be interesting $\mathcal{D}$ will usually have a structure relevant to the interpretation of the expressions in $\mathcal{S}$. It is common to take $\mathcal{D}$ to be a set of possible worlds, of which one—the so-called *standard interpretation*—is assumed to be the real (actual) world, and where worlds are taken to consist of a set of objects, properties, relations, facts and/or propositions, possibly states of affairs (objects exemplifying properties and standing in relations), etc.—i.e., to exemplify that which in §6 I call *classical ontology*.
2. Relations:
   a. A *derivability relation*, $\vdash$, instances of which hold between one or more expressions $\sigma_1, \sigma_2 \ldots \sigma_j$ of $\mathcal{S}$ and another element σ^* of $\mathcal{S}$;
   b. An *interpretation function* I, maps elements of $\mathcal{S}$ onto elements of $\mathcal{D}$; and
   c. An *entailment relation*, $\models$, instances of which, like $\vdash$, hold of one or more expressions $\sigma_1, \sigma_2 \ldots \sigma$ of $\mathcal{S}$ and another element σ^* of $\mathcal{S}$, just in case, in the semantic domain $\mathcal{D}$, the interpretation $I(\sigma^*)$ is a *semantic consequence* of the interpretations $I(\sigma_1), I(\sigma_2) \ldots I(\sigma_k)$. A standard notion of semantic consequence is that $I(\sigma^*)$ be true in every possible world in which the interpretations $I(\sigma_1), I(\sigma_2) \ldots I(\sigma_k)$ are all true.

Thus in a particular case '$\textsc{man}(\textsc{socrates})$', '$\forall x\ \textsc{man}(x) \supset \textsc{mortal}(x)$', and '$\textsc{mortal}(\textsc{socrates})$' might be elements of the syntactic domain $\mathcal{S}$ (call them σ_1, σ_2, and σ_3, respectively). And, crucially, the real historical person we call Socrates would be an element of the semantic domain $\mathcal{S}$, along with the abstract properties of being a man and being mortal, and various facts about who exists, who is a man, who is mortal, etc. (with '$\textsc{socrates}$' mapped by $\mathcal{I}$ onto Socrates, '$\textsc{man}$' onto the property of being a man, and '$\textsc{mortal}$' onto the property of being mortal). Thus, since in all worlds in which the interpretation $\mathcal{I}(\sigma_1)$ and $\mathcal{I}(\sigma_2)$ are both true, so as well is the interpretation $\mathcal{I}(\sigma_3)$, then we would say that σ_1 and σ_2 *entail* σ_3, or as it is written, $\sigma_1 \wedge \sigma_2 \models \sigma_3$ (i.e., in any world in which Socrates is a man and all men are mortal, Socrates is mortal). In addition—a separate fact—σ_3 can be formally *derived* from σ_1 and σ_2, written $\sigma_1 \wedge \sigma_2 \vdash \sigma_3$.

I have said that there is also a norm. The point is simple: one defines a logic in which it is presumed that what is derivable should be what is true or entailed. Although one could formally define spectacularly many logical variants in which the derivability relation parted company completely with the entailment relation, *one would not*. In typical mathematical treatments, one simply defines *soundness* (that what is derivable is entailed) and *completeness* (that what is entailed is derivable) as abstract properties of the system, and then tries to prove them true, or anyway demonstrates that the system is sound (else it will be discarded or redefined, or anyway dissed), and attempts to see whether it is complete. While on the surface these practices may not look normative, in point of fact, I believe, they are manifestly norm-driven.

Fundamentally, the norm has the following structure:

> **«Name»**: *What can be done, effectively, should honour the semantic, which in general will not be effective.* ⟦possibly increase re paucity of effective, reach of semantic—towards the Representational Mandate?⟧

In my view, this same norm motivates computing, and human thought.‡

†The names $\mathcal{S}$, $\mathcal{D}$, and $\mathcal{I}$ in this exposition are my own, as is the use of σ (σ_1, σ_2, σ_i, etc.); the symbols '$\vdash$' and '$\models$' are standard.

‡For more details see "Representation and Registration," ch. ■n of Volume II.

ties in intension, states of affairs, or what some philosophers call "tropes": objects exemplifying properties, as typically denoted by the use of gerundial phrase in English, such as *being a mesoscale projectile*, or *weighing 120 pounds*. Thus a pulse on an electrical line may be effective (capable of turning on a switch, say), but not merely in virtue of being an identifiable object—say, being the pulse named 'pulse$_{723}$', or being "the pulse referred to in this paragraph"—but because of *being an electrical pulse of a certain voltage*. As the object that it is, the pulse I am calling pulse$_{723}$[60] will exemplify an indeterminate number of other properties as well, such as perhaps occurring exactly four billion seconds after Alexander Graham Bell cried "*Mr. Watson, come here! I want to see you*!", or being the pulse that I was hoping we would see be emitted by a pulse generator this afternoon in the lab, or being the pulse under discussion in these pages, etc. Its being effective at turning on the switch, however, results from its exemplification of the former property, not from its exemplification of any of the latter ones.

The question on the table, therefore, is about what properties in a system of formal logic are required to be effective or mechanical—and/or, which may be the same question, what relations required to be effectively "computable," in the standard sense. And given the foregoing preliminaries, on at least on the default realist position, I take it that in classical logic the answers would be taken to be unambiguous:[61]

60. There is no problem using the name, so long as we know that the pulse's efficacy as a pulse does not derive from being so named.

61. It is not as if there is a universally accepted definition of exactly what it is to be a logical system, and someone might object that they could define a system of logic in which the syntactic properties were not effective, or derivability was not computable, etc. But if one takes leave of these constraints, it becomes possible to vitiate all standard logical results. For example, one can violate Gödel's incompleteness results for arithmetic by defining the property of being true in the standard interpretation to be a *syntactic property* (at which point it becomes trivial to define an axiomatisation that is sound and complete).

1. *Syntax* must be effective, in the sense that the exemplification of syntactic properties by expressions in $\mathcal{S}$ must be effective (for example: it must be mechanically determinable whether or not an arbitrary expression σ_i exemplifies an arbitrary syntactic property $\mathcal{S}_j$); and
2. *Derivability* must be effective, in the sense that the relation $\vdash$ must be effectively computable.

If one were to define a "logic" that violated either of these two principles, it would thereby be evacuated of intellectual substance and all theoretical interest. For example, suppose one were to stipulate "true in the standard interpretation" to be a *syntactic* property of arithmetic sentences, and to claim that β could be derived ($\vdash$) from α just in case β is true in any world in which α is true. Then, contrary to Gödel, common sense, and all that is right and good, one would perforce immediately have a sound and complete axiomatization of arithmetic.[62] That is, to put it bluntly, in the development of a logic, to violate either of the above properties is to *cheat*.[63]

On the other hand—and this is what matters to my stance against blanket mechanism—there is no reason whatsoever that the other elements be required to be effective, or be restricted to the realm of mechanism:[64]

1. The semantic domain $\mathcal{D}$ itself (i.e., the exemplification of properties by objects in $\mathcal{D}$ denoted by expressions in $\mathcal{S}$); or

62. «…???…»

63. The implicit rules in logic and various allied technical subjects, contravention of which constitutes cheating, are a gold mine of insights about what is genuinely going on under the surface. In AOS I explore the conditions on what is known as a "reasonable encoding" as a device to probe the metaphysical presuppositions and fundamental subject matters of computability theory. See especially AOS Vol. III, and also ch. 1 of the present volume, "The Foundations of Computing."

64. «Except refer to Earman.»

Reference

Semantics, especially in logic, is often thought about primarily in terms of *truth*—the semantic value, it is claimed, of sentences and claims. But *reference*, in my view, is equally if not more important (notwithstanding Frege's insistence that reference serve as the handmaiden of truth): the semantic value of names, noun phrases or terms. Roughly, the reference relation is what holds between a name, such as 'Stravinsky,' and the person so named (the Russian-born composer who died in 1971)—or between the thought processes of a lover and the person whom they love, be they immediately in the vicinity or a thousand miles away. What reference makes evident, but in fact is equally if less blatantly true of sentences, is that semantics involves relations between linguistic or symbolic items or expressions and *a wider world*.

What matters in discussions of the relation between blanket mechanism and critical mechanism is that semantics, in the sense that underlies reference, is not an effective relation. Semantics is not an *activity*; nothing *happens* in order to connect an utterance of 'Stravinsky' with the long-dead composer. By analogy, consider the property of being the tallest person in Milwaukee, or being one hundred miles from Vancouver, or having a hairdo reminiscent of Lyle Lovett's. Such properties are not *effective processes*; they merely *obtain*—they hold, from time to time, in this or that location, of this or that entity.

Sometimes, in teaching introductory cognitive science, I explain to students that from time to time they exemplify the property of *being thought about*—by their friends, their parents, their teachers. But nothing effective happens, in or on or around them, that can be detected, when this property is exemplified. At one moment they are "thought about"; at another, they are not. Yet no engineer will ever build a device (an iPhone app, say) that can detect the fact—since no discriminable effective signal distinguishes the two cases.

In response, a surprising number of students, under the spell of what I am calling blanket mechanism, start to deny that they have been referred to, after all. Because being referred to cannot be detected, *it must not happen*, they reason.

… Talk about "speed of logic", ref Church at CSLI, *etc. …*

2. The interpretation function I; or
3. The entailment relation $\models$.

In fact it would be perverse to assume that any of the last of these three entities was subject to effectiveness constraints.[65] Logic is undeniably powerful, on the one hand, and also strikingly, and famously, limited, on the other—and interesting because of both the source of the power and the reason for the limitation. And both arise, metaphysically, from the same fact: by restricting oneself to the effective or mechanical, one can achieve extraordinary results as regards reasoning and proof about a *much, much wider world*—specifically, a world that is *not so restricted*.

In contrast to the blanket variety, I will use the term **critical mechanism** to label recognition of this fact that although $\mathcal{S}$ and $\vdash$ *are* required to be subject to constraints or effectiveness or mechanism, $\mathcal{D}$, I, and $\models$ are *not* so restricted—or more generally, the recognition that although thinking and reasoning must, in an appropriate sense, be mechanistically implemented, it is nevertheless no part of an overarching physicalist worldview to require that the world itself, or semantics (i.e., the world *reasoned or thought about*) be subject to such constraints of mechanism, effectiveness, or computability.[66] I take

65. See the sidebar on reference.

66. To assume that physicalism implies blanket mechanism is simply a mistake. Among other things, it runs roughshod over differences between what is global and what is local, and presumes the strongest sort of type reductionism (cf. Fodor's discussion of the "Special Sciences"—«ref»). But even aside from those issues, consider relativistic strictures against travel faster than the speed of light, and think about the property of being referred to. As noted on page ■n, Andromeda can be thought about *today*, in virtue of an action we take here on earth, even if it would take 2.6 millions years for an effective signal to travel from here to there. Or even more simply: consider the property of *being* 2.6 millions years from Andromeda. That is a real property exemplified by the world, but it is not an effective property. No one could make a switch turn on in

Evolution

In the debate with Daniel Dennett reprinted in Volume II, an issue comes up about why, in *On the Origin of Objects* and "Rehabilitating Representation,"† I did not discuss evolution. The discussion there illustrates a divergence of views that resembles, if it is not in fact, that being drawn here between blanket and critical mechanism. If I understand him, Dennett believes not only (i) that phenomena such as human representation evolved though processes of natural selection, but also (ii) that, perhaps because of that fact, they are to be explained in evolutionary terms.

I believe (i) as well. I do not endorse Intelligent Design; I do not think that our capacities are magic. But as I say in those pages, just because evolution is the way in which our representational capacities *arose* does not mean that is what representational capacities *are*. Rather, I believe that representation is a way of being that is possible, in the world—and that evolution, as it has in so many other cases, came across it (presumably at random), and, because it conferred such a stunning advantage on creatures, exploited it hugely.

The relation to blanket mechanism is as follows. It may be (presumably is) that evolution is the *mechanism* by which we came to be as we are, just as our brains (and bodies and societies) embody a mechanism with which we understand the world—and just as a calculator, too, embodies a mechanism, one that responds to the structures and patterns of various numerals and buttons and other internal effective configu-

critical mechanism to be the view that historically motivated and underlay the development of logic, at least in the main.[67] Critical mechanism is also close to my own picture of what the world is like, overall, and of the place of mechanism (or the effective) within it.

To deflect potential misunderstanding, I should say that I

virtue of its instantiation.

67. «cite formalism and strict mathematical constructivism as a possible counterexample; but not intuitionism»

rations so as to produce arithmetically intelligible behaviour. That is: there is an account of how the calculator works; there is an account of how we work; there is an historical account of how evolution worked. But just as a mechanical account of how a calculator works is just half of an account of it as a calculator, and not, as it happens, the half that explains how what the calculator does is *arithmetic*—and just as a syntactical account of how a theorem prover works is just half of an account of it as a theorem prover, and again, not an account of how it is that what the theorem prover proves are *theorems*—so too the causal account of the human brain and body, and the evolutionary account of human biology, are just half accounts of us. But in all of these cases, because the systems in question are intentional, the mechanistic accounts say, perhaps even explain, how the system in question works, but they do not explain *what the systems are*, or *what they do full bore*. Answering those questions requires adversion to semantics—and semantics in turn requires adversion outside the realm of mechanism. The fact is clear, and well recognized, in the case of calculators and theorem provers. So too, in my view, in both aspects of the human case. An account of the fact that we *think* requires adversion to more than the brain. And an account of how we got here requires adversion outside of the (syntactic, or at least mechanical) realm of evolutionary selection.

†An earlier version of "Representation and Registration," also included in Volume II.

am by no means a naïve realist, as will be briefly discussed in the next section, and anyway is evident from my *On the Origin of Objects*.[68] So I am not embracing the default realist position alluded to above. Rather, in embracing critical mechanism I am saying that there is *something deeply right and hugely important about a realist reading of logic*. To say what it is that is right about it, in ways compatible with my own metaphysical

68. «Ref?»

position,[69] I will employ terminology in the way I used it in that book: using *ontology* to name the *world-as-registered*—i.e., to name the objects, properties, relations, states of affairs, etc., that we intelligibly find to be in or to constitute the world—and *metaphysics* to name that which we so find intelligible. In these terms, I take the *metaphysical* world to be vast, unutterably large, defiant of description, and drenched in that which we register as an infinite number of object and properties and relations and the like. The *ontological* world—the metaphysical world as registered—I will take, at least for present purposes, to consist of the plenitude of objects, properties, relations, types, facts, and other denizens of reality in terms of which we find the metaphysical world intelligible.

Given this distinction, the point about the mechanical or effective is this: of the still-surpassing abundance of ontological richness in the world (even if that pales in comparison with that which it registers), only a tiny subset—a subset of measure zero, to use the set theorists' phrase—meets the conditions of being mechanical or effective. Or to put it in plain language: *almost nothing is effective*. Being the shirt my grandmother sewed for me, being a thousand miles from Abilene, being about to receive a letter from my long-lost friend Akiko—none of these things are effective, none are of the sort that could be harnessed to turn on an electric switch. And to take another example relevant to the essays in this volume: so too is *being referred to* not an effective property.[70] It is especially true that *being referred to right now* is not effective (Andromeda, for example, at this very moment enjoys the property of being so referred to, at least in thought, since I am thinking about it, in spite of being more than two million light years away—and in spite of the fact that no evidence of the fact that that it is currently being thought could reach Andromeda until two million years in the future). In fact it is metaphysically astonishing, to say nothing

69. «Refer to vocabulary discussions in AOS—radical, reactionary, etc.»
70. Again, cf. the sidebar on reference, p. ■n.

of being deucedly lucky, that *anything* is effective—that it is in virtue of the exemplification of any property that anything can have any causal consequence whatsoever.

The astonishing paucity of the causal or effective, I take it, establishes a ferocious challenge for humans, for logic, and for computation. It is a challenge that ultimately derives from physics.[71] If anything is to *happen*, it must happen, as we often say, *causally*—must happen in virtue of the exemplification of causal, or mechanical, or effective properties (which may all be the same, may be different; I am speaking at a very high level of abstraction for the moment, and anyway no one quite knows what the differences are, if any). This is just a blunt, inescapable fact about what the world is like. Fundamental physics is an account of what can happen at the microscopically small level. It may be—I will have much more to say about this in due course—that computer science is relevant to an analysis of what can happen at arbitrary levels of abstraction, at arbitrary levels of grain. We are not yet sure, because we do not yet know what computer science is. We do not know, that is, once we let go of the conceit that the 'c' word names a theoretically interesting, delineated subject matter or natural kind, and incorporate its insights into our general intellectual world view, what those insights will turn out, all this long while, to have been about. But however that issue about computer science goes, the following is the bottom line: *what happens* does so in virtue of the mechanical or effective, but *what is the case* is stupefyingly larger.

Moreover, if I can put it this way, the challenge of *knowing*—of reasoning, of computing, of being able to think—de-

71. I am not a straightforward physicalist, either—even of the weakest form so far articulated: global supervenience. But just as I believe that there is something profoundly right about realism, which must be preserved in any more constructivist or embodied alternative, so too I believe that there is something both incredibly sobering and yet surpassingly powerful about physicalism, which must also be, if not preserved, then at least done justice to in any successor or alternative account.

rives from the discrepancy between the vanishingly small and restricted subset of the world that is mechanical or effective, and the vastly larger world of, to channel Wittgenstein, that which is the case. Knowing has to happen, reasoning has to happen—we do not come blessed with divinely instilled comprehensive knowledge. What we humans have succeeded in learning, by evolutionary and then societal means, is how to exploit that which is mechanistic in ourselves, and in our environment, in ways that allow us to stand in referential (if not always reverential) relation to the world as a whole. An ability to appreciate the magnitude of this achievement is what I take to be so important about critical mechanism. It is also what I take formal logic to be a magnificent first stab at explaining. And this, too, is what computing is a radically more complex practice exploring. That it is metaphysically possible at all, as I have already said, is both amazing and fortuitous.[72] That humans have evolved so as to be able to do it is undoubtedly our most staggering achievement. That we are slowly coming to understand what it involves, and, Golem-like, in systems of our own devising, are starting to construct synthetic instances of it—well, let me just say that that is what I think it is that makes computing important. It is a sign that we are entering a new stage of development of unutterable historic consequence.

As should be obvious, these are all points on which the blanket mechanist is sentenced to blindness: that this is the nature of knowledge, the challenge of being, the importance of logic, the character of computing. Blanket mechanism assumes that the *world* is restricted to the effective. Critical mechanism leaves the world whole and vast, and realizes that all that is so restricted is *what happens*, in a very particular and limited sense of what happens.

72. There is an anthropic principle worth a pint: what the chances are—what the requirements are—that the universe provide the capability for creatures made of clay to know it.

These volumes are not a metaphysical defense of critical mechanism, nor a systematic account of the nature of computing or reasoning in its terms. They are merely a collection of papers that are small steps en route to such a story. But the papers were knowingly, if implicitly, written within the overarching framework of a critically mechanical viewpoint. Because of this, the individual moves I make in them—down to the most intricate details of 3Lisp programming—can only be understood against its strictures. Blanket mechanists, I know, found 3Lisp inscrutable when it was first introduced—and will likely still find it inscrutable today. I only hope that these few introductory remarks will help to make its architecture, and the concerns of the other papers in these volumes more broadly, just that little bit more comprehensible.

6 Metaphysics

Fourth and finally, a word about a gradual shift in the topics these papers address, starting in this volume and continuing through Volume 11. As explained above, I started out in the 1980s with a relatively restricted focus on the nature of computing and the semantics of computational programs and processes. Gradually, the concerns evolved to include unrestricted issues in semantics and ontology. This shift from an inquiry into the foundations of a particular "science" into a full-blown metaphysical investigation had already started to take place before I came to the "laboratory, not subject matter" conclusion about computing discussed above in §3. In order to do justice to real-world computing—what in 03 and A0S I call *computation in the wild*—I was more and more driven to consider the nature of the world in which computing is found, and to which it relates.

In the first instance that focus grew out of straightforward semantical requirements. If programs or processes are to be theorized as intentional—as signifying, representing, modeling, or simulating or in some other way as being "about" the

world—then, in order for relations between the two to be brought into account, one needs a theoretical framework for talking about the world towards which they are directed.[73] But as usual, what initially came into view through an interest in semantics soon broadened into a more general concern. In what becomes a pervasive pattern in these pages, the broadening takes place in three discernible stages. First, because the variety of types of semantic relation between computational process and task domains to be analysed seemed almost arbitrarily unrestricted, doing justice to the richness of practice re-

73. See the sidebar on "The Semantics of Programs and Processes" on pp. lvi–lvii.

The Semantics of Programs and Processes

Not everyone takes programs to be about the world. As explained in some detail in Part B, it is standard in computer science to take the subject matter of programs, taken as semantic entities subject to semantic interpretation, to be *the computational process or behaviour that results from executing them*—a view that in this volume I refer to as a "specificational" view of programs. The approach adopted in 3Lisp, and on which the model of reflection presented in the chapters of Part B is founded, differs in taking the semantic domain of the program to be *the task domain over which the behaviour is defined*—an approach that I instead call an "ingredient" view of programs.

If one adopts the more traditional specificational view, one might label the relation I was interested in, between the computation and the (typically external) world or task domain, as *process semantics*, in the sense of being the semantics of the semantics of the program-viewed-as-specification. As usual, however, one confronts the terminological issues discussed in §4. In computer science, the term 'process semantics' is used to denote theoretical models of the *behaviour* of typically parallel and/or communicating systems. Why this is called 'semantics' escapes me, though one can understand it as being an analysis of the semantics

quired broadening the analysis from *computational semantics in particular* (i.e., as if, given a reliable general notion of semantics, the task was to discern the nature of the computational species) to *semantics in general*. Setting specifically computational considerations aside, therefore, I was drawn to investigate, in the general case, fundamental notions of reference, representation, meaning, content, specification, instruction, and the like. Second, it became increasingly clear that semantics is far from the only place metaphysical challenges arise. As suggested in the "ontological wall" phrasing of chapter 1, doing justice to the nuances of computational phenomena themselves also outstrips the capacities of received frameworks. Quite apart from

of programs for parallel and/or communicating processes. The point, though, is that what is mathematically constructed, as is typical of model-theoretic technique, are structures that correspond to the computational behaviour itself—not to any world or task domain towards which that process is directed, or that the process is in any sense "about."

The fact that computer science has 'used up' semantical vocabulary to talk about "computationally internal" relations is presumably what led Allen Newell, once the importance of the relation to the world was brought to the fore, to dub a view of programs that does treat real-world relations as "The Knowledge Level." †

However one labels it, that was the relation I was targeting, in analysing the semantics of computing: the relation to the external world, to the real-world task domain that computational processes are about, to the world about which "data" structures contain data So having a workable framework in which to talk about the world was theoretically requisite.

† The term was introduced in Newell's Presidential Address to the American Association for Artificial Intelligence (AAAI) in 1980. (Cf. also my own reply to that paper, written while I was still a graduate student (Smith 1981). «say something about it»

issues of what computations signify,[74] in other words, simply describing the programs and structures and processes that do signifying led me into full-fledged ontological investigations.[75] And then the third: studying computational reflection (the topic of the parts in Part B) brings the two topics together, in concentrated form.

The change in focus from computing to metaphysics, reflected especially in the last two papers in the first volume[76] and most of those in the second, involves not just a shift in subject matter, but also the mounting of a critique. In particular, two standard, related background metaphysical themes come under increasing fire—themes that I had unquestioningly assumed at the beginnings of my investigations:

1. What for discussion I will call **classical ontology**: the world as imagined in the traditional image of discrete objects arrayed in (what at least philosophers would assume to be) the usual way: exemplifying properties, standing in relations, grouped together in sets and states of affairs— possibly with positive and negative

74. As should be obvious (from 03 and elsewhere), I do not believe in any such presumptuous ontological/representational divide. The point is only that even if one were to endorse such a strategy, the ontological structure of computing itself—what programs are, what makes one program one and not two, the nature of data structures, how to analyse implementation relations, etc., demand more than what classical ontology can provide.

75. The details of the inadequacy of classical token-type (or class-instance) frameworks to accommodate the complexity of real-world identity conditions and layers of abstraction are not much explored in the papers in these Volumes. Some discussion of a more flexible alternative lurk in the pages of 03; see for example section «ref layers of particularity».

76. "Linguistic and Computational Semantics" and "The Correspondence Continuum," chs. 10 and 11 of Volume 1.

polarities,[77] etc.

2. Something like a **correspondence** theory of semantics, in which representations are taken to correspond, in a relatively straightforward compositional way, to the states of affairs in the world they are about, along roughly the lines of Wittgenstein's famous "picture theory" of language or semantics.[78]

Both themes are entirely familiar Both are almost universally assumed in model-theoretic and other formal analyses of the semantics of logical formalism. And both, I argue in these papers, are inadequate bases on which to explain real-world computing.

Now it may seem perverse, in these first few decades of the twenty-first century, to spend any time at all developing a cri-

77. E.g., a state of affairs consisting of objects α, β, and γ, with α exemplifying the positive property of being a chair, β of being a table, γ of being a cup, and γ also exemplifying the negative property of not being on β.

78. The idea is not that thoughts or representations are necessarily like pictures in the mundane (illustrative, two-dimensional) sense of that word, but rather that words correspondence, piecewise, to facts or propositions in the way in which (mundane) pictures are thought to correspond, piecewise, to the situations they depict. Rather—setting aside myriad nuances, complications, and exceptions—the idea is roughly this: complex terms (referring expressions) are taken to represent ontological entities or states of affairs, directly or indirectly, according to a scheme in which the represented ontological structure is systematically specified in terms of the grammatical structure of the representing complex, with each component of the grammatical structure recursively representing a corresponding component of the represented ontological structure, according to a recursive application of the same overall scheme.

In many correspondence theories facts or states of affairs are taken as the *truth-makers* for sentences or propositions, where truth serves as the over-arching norm in terms of which the semantics is framed. But correspondence theories are not restricted to sentential representational schemes; term arithmetic can be given a correspondence-theoretic semantic account, even if it does not support the making of *claims*.

tique of what looks for all the world like an excessively familiar pastiche of naïve realism, context-independent semantics, etc. That picture of the world, something it may not be too distracting to call *modernist*, has been so thoroughly subjected to withering attack, over what is now many decades, that some readers may doubt that it any longer deserves serious attention. It may seem especially absurd to criticize the classical picture in the ways that I do here—especially in "The Correspondence Continuum," the final paper of Volume I.

The concern has some merit. I myself sometimes feel, when trying to explain the labyrinthian contours of the argument in that paper, as if I am leading the reader on a tortuous path, hacking a rough-hewn trail through dense thicket with machete and crowbar, only to emerge at the end into a clearing that, lo and behold, is not only already well populated, but could have been much more easily reached by a perfectly serviceable road—a well-used road of long standing, which travels via an alternative, and much easier, route.

But I do not think the situation is that simple. And it is that which warrants a word of explanation here.

It was once an informal truism that computer science was—perhaps among other things—a study of complexity. That description was widely bruited in the 1970s and 1980s, in the corridors of computer science and artificial intelligence laboratories, prior to the development of a number of more specific subjects eponymously labeled as "complexity theory." Over the last decade or two, in particular, the term 'complexity theory' has come to have two specific uses. First, it is used for what is also called the "theory of complex systems"—a theory based on self-organizing processes, complex adaptive systems, etc., emerging in part from what was once called chaos theory, was soon relabeled non-linear dynamics, and now pretty much goes by the label "dynamics," and which theorizes attractors, criticality, bifurcations, and so on—as made famous not only

in popular science writing[79] but also by an upsurge in interest in such self-organizing systems widely associated with the Santa Fe Institute. Second, as mentioned in §2, above, within computer science the term 'complexity theory' is also used to name a mathematical enterprise also called "computational complexity theory," equally technical, and briefly discussed above, which studies the "difficulty" of running algorithms. But independent of and prior to these more technical studies, the more general use arose out of a simpler recognition: not only that computer systems are or at least can be stunningly complex in mundane terms,[80] but also that expertise in computational matters involves developing expertise in understanding, building, managing, and theorizing systems of such complexity. This image was among other things advanced in Simon's groundbreaking *The Sciences of the Artificial*,[81] in which the no-

79. «Refer to Gleick's book»

80. While detailed calculations are otiose, it is fair to say that current computational processes have computationally relevant features spanning more than a dozen decimal orders of magnitude in both space and time. The switching time of contemporary processors is on the order of ten picoseconds (10^{-11} seconds), and the runs of discrete programs—i.e., programs designed to start and then stop, rather than continuing processes, such as the internet and web sites—are measured in up to days, (10^5–10^6 seconds). This yields a temporal range of events of interest to computer scientists of $\sim 10^{16}$. Spatially, prototype circuits have been sampled on $\sim 10^{-8}$ meter processes (minimum feature size of about ten nanometers), and server farms built occupying large buildings of, say, $\sim 10^2$ meters linear dimension, for a one-dimensional span of $\sim 10^{10}$, or in the relevant two dimensions, $\sim 10^{20}$.

In saying that these dimensions are "computationally relevant" I mean that structures and events at this range of scales are all *analysed in computational terms* or *exhibit computational regularities*, rather than, say, merely being physically analysed—e.g., in physical or biological terms in ways that subserve processes that neurological or psychological in nature (subject to neurological or psychological regularities).

81. Simon, Herbert, *The Sciences of the Artificial*, MIT Press, 1969; third edition 1996. Complexity was a major theme of this work, as reflected in the titles of the last two chapters: "Alternative Views of Complexity" and

tion of complexity plays an especially prominent role.

So computational phenomena are complex. In addition, and again among other things, computer science is also an engineering discipline. Systems are built; programs written; processes materially unleashed. It is not enough, within the context of such an engineering practice, for accepted theoretical frameworks merely to be descriptive—not enough for them to characterize their subject matters in broad brush terms. Rather, frameworks adequate for backing engineering must be specified in *sufficiently precise detail to enable complex, concrete construction*. The complex, concrete actuality of real-world computational systems is a fact of extraordinary methodological import, whose theoretical importance cannot be overemphasized. It plays an important role even in my more foundational AOS work, by for example buttressing arguments that, in computational guise, notions of *syntax* and *form*, far from being abstract in the way in which Searle imagines in both of his arguments against the possibility of genuine artificial intelligence,[82] are on the contrary fundamentally concrete—grounded in very real physical constraints. But the actuality of computation also bears on the present issue, of the two-pronged critique of classical ontology and correspondence theories of semantics.

The point can be formulated in terms of a striking opposition. On the one hand is the widespread critique of the classical or modernist view already mentioned, which for discussion I will loosely here call the **discursive critique**, which has been widely advanced over the last hundred years in quarters as diverse as Wittgensteinian philosophy, Heideggerian phenomenology, poststructuralism, cultural theory, feminist epistemology, situated cognitive science, and science studies. At the same time, strikingly, and over roughly the very same hundred years, a stunningly complex technical infrastructure has

"The Architecture of Complexity: Hierarchic Systems."

82. «ref».

developed (to use the term 'infrastructure' broadly, e.g., in the sense of Star & Bowker[83]), marinating contemporary life in a world of computation, information, communication technologies. Yet, I believe it is fair to say—and this is what commands attention—the discursive critique has to date had little to no effect on the development of that infrastructure, at least not on the theoretical frameworks that underlie its foundational constructions. From the point of view of the plumbing, it is as if a century of discursive philosophy never happened. The semantic web, contemporary specification languages, metadescription frameworks, and the like, all remain formulated in entirely classical or modernist terms.

This discrepancy generates the following obvious question: *Why has the discursive critique had so little influence on contemporary technical infrastructure?* And, less obviously but perhaps more revealingly: *Are computationalists—programmers, engineers, computational theorists, etc.—immune to its considerations?*

A flood of possible answers come to mind, whose sheer variety, let alone relative importance, are difficult to assess. Thus consider one potential factor that some will cite: conservative market forces, based on underlying neo-liberal assumptions of contemporary capitalism. There is likely something right about this. It is certainly true that, as yet another consequence of being actual, our contemporary socio-technical infrastructure has grown inextricably enmeshed in powerful networks of finance, power, politics, vested interests, and the like. It is also my experience that a majority of working programmers feel that the our embeddedness in this infrastructure is already so deep as to make discontinuous or even disruptive change impossible. The only route forward, such people argue, is via slow incremental adjustment, subject to the double constraints of commercial viability and state sanction. Sure enough, it will

83. «ref *Sorting Things Out*».

be admitted, there was a time, in the heyday of Silicon Valley during the 1970s and 1980s, when the whole heady affair was up in the air—but that time is no longer.

I confess that I am unclear on how deep this argument penetrates, however. Even if *discontinuous* changes are unlikely, it would be absurd to deny that *profound* change not only remains possible, but is taking place on our very watch. As is increasingly widely decried, consolidating financial interests, nationalist agendas, and other large-scale political forces appear to be gaining an inexorable (and hugely worrying) grip on matters of control, surveillance, privacy, autonomy, intellectual property, freedom of speech, and the like.[84] "Information wants to be free!," Steward Brand famously said, less than thirty years ago[85]—at a time when the internet was commonly viewed as being an almost deterministic instigator of increased democracy. Yet today rising cybercrime, government surveillance, abrogrations of privacy, etc., make such naïve or perhaps libertarian optimism seem almost quaint. I am far from being a political or social theorist, but one would be a fool not to recognize the continuing power of socio-political and economic influences.

Yet these changes seem almost eerily unrelated to the question at hand, of why considerations from the discursive critiques have not penetrated the theoretical scaffolding underlying our technical infrastructure—at least not in evident ways. As has already been mentioned, not only presently existing but even emerging new standards for identifying resources on the web—URLs, URIs, document object identifiers (DOIs), etc., plus OWL, RDF, and other description languages and frameworks—are all based on, or at least compatible with, what is known as "common logic," a relatively weak fragment of classical first-order logic, understood in ways that directly reflect

84. «refs; Diebert? Castells? Benkler?»

85. «ref; and acknowledge that its fame derives from John Perry Barlow's quotation»

what I have identified as our two modernist presumptions.

I believe there are deeper reasons why the discursive critique has yet to exert a decisive influence on the technical underpinnings of computing—reasons that relate to why it remains important to analyse the modernist approach in a level of technical detail to which, to my knowledge, it has never been subjected.

The first has to do with the relentless specificity and wealth of analytic detail that needs to be explicitly articulated in order for us to see how to move forward towards a practicable alternative. No matter how successful they may seem on their own terms, nothing in any of the existing discursive critiques has been worked out at anything even approximately the level of precision and thoroughness that would be required to underlie systematic computational *synthesis*. Even though from some perspectives one might view the fact as unfortunate, it is nevertheless no accident that computing itself, and computer science's entire edifice of concomitant theoretical scaffolding, emerged from the Anglo-American philosophical and metamathematical tradition—rather, say, than from pragmatist semantics, continental philosophy, or any other interpretive, literary, or discursive tradition. The technical demands of providing an intellectually comprehensible conceptual schema in terms of which to construct systems with millions or billions of interacting constituents are immense—vastly greater, in my experience, than those without programming experience generally realize. Even technical prowess in philosophy and logic do not prepare one for this insight. Think of various axiomatizations of modal and deviant logics, such as S5 or $\blacksquare_n$, which typically consist at most a dozen axioms, and compare them to programs consisting of millions lines of C++.[86] The differences are great enough as to constitute a difference in kind,

86. See "A Hundred Billion Lines of C++," included here as Chapter 8 of Volume I.

not merely in quantity or degree. This is why , in teaching I regularly claim that no one has written a program until they have written a program at least 20,000 lines long.[87]

That then is the first point: modernism will reign as the foundational theory of computing, in my view, until someone provides an alternative that is capable of serving as a framework for this order of detailed, concrete, synthetic construction. Nothing in the discursive critiques has yet approached that status. From a programmer's perspective, phenomenological or Wittgensteinian or poststructural or feminist critiques seem unremittingly metatheoretic—disquisitions on the character a better theoretical edifice would have, rather than such a theoretical edifice itself.

The second reason for conducting a detailed analysis may be more interesting. One thing that emerges, from close inspection, is that contemporary practice is far less modernist than one might expect, given the state of reigning theoretical frameworks.[88] Discursive critics will not be surprised, but it is again the details that matter. In fact—and this is one of the most important themes pervading these papers, something of a major plot line, though it does not receive explicit attention except in the introduction to the last paper in Volume 1—many of the oddities, missteps, and confusions that I "diagnose" in these papers, including the fundamental semantical "errors" of which I accuse Lisp, can be more charitably (and ultimately, I believe, more fairly) understood as efforts, on the part of pro-

87. Novels, series of novels, whole literatures—these are literary constructs, which discursive traditions analyse, to which phenomenological analyses may be deemed appropriate. Perhaps, some day, we will have technical scaffolding in place so that one can construct programs in ways analogous to how we currently write novels. To get there, though, we are going to have to develop infrastructure with complexity approaching that of the human brain—to say nothing of understanding how such infrastructure works.

88. «Ref Latour's *We Have Never Been Modern*»

grammers saddled with modernist scaffolding, to sidestep the inadequacies of that very framework, in order to accommodate some of the very sorts of phenomenon that have been so forcefully brought to our attention by the discursive critique.

Compositional semantics is a case in point, to which it is instructive to devote a moment's attention.

The originating intuition behind the idea of compositional semantics was roughly this: a complex representation, such as a sentence, can be semantically interpreted "piece-wise," with nouns denoting objects in the domain of discourse (the subject matter being talked about), adjectives and verbs denoting properties, etc. So in the sentence "My Bernese knocked over the table again," the term 'My Bernese' would denote a dog, 'the table' a table, 'knocked over' a dynamic interaction between them, etc.; and the whole complex sentence would be *true* (would "denote the truth," if one's theoretical predilections run Fregean) just in case the denoted dog did in fact interact in the signified way with the denoted table. That is: compositional semantics, in its originating framing, was decisively an instance of a correspondence-theoretic approach.

Programming languages are universally interpreted (semantically[89]) in terms of compositional semantics. On the other hand, most programming languages contain constructs that defy any simple conception of correspondence. Consider a construction such as `(ABORT!)`, the execution of which causes the entire program in which it occurs simply to come to a crashing halt—or the slightly more complex `(RETURN α)`, structured in such a way that the value of α is returned, not simply as the value of the particular expression within which it occurs, but returned as the value, again, of the entire encompassing process in the course of which execution of the particular expression was triggered.

Ever ingenious, computational theorists have devised com-

89. «As discussed in …, the term 'interpretation' is used in computer science for … »

plex ways to preserve compositionality so as to accommodate disruptive or "non-local" constructs of this sort. The fundamental move is to abandon the idea that the "denotation" of any particular element, or at least what the semantic interpretation function maps the element onto, is what, intuitively, that construct stands for or names. By doing this, it is straightforward—for example by using what computer scientists call a "continuation-passing" semantical model—to arrange things so that effectively arbitrary long-distance, non-local semantical facts can be accommodated. For example, in the 1980s, when explaining 3Lisp to the late Jon Barwise, I devised a variant of the λ-calculus extended with a special operator '✶', so that, modulo some subtleties,[90] the term '✶' would be the result of normalizing any complex expression, no matter how large, in which that form occurred.[91]

The point generalizes. If any theory or account θ can be given about the interpretation of whole texts—be that theory context-dependent, invoking consistency instead of correspondence as the fundamental semantic norm, customizing interpretation by the contingent circumstances of the reader or particularities of history, incorporating holistic facts about discursive context, or whatever—then a "compositional" semantics for the individual terms within the text can still be constructed such that the interpretation of the whole text will be as specified by θ. Suppose in particular that τ is a text in a functional language Σ[92] comprising expressions of various

90. «discuss: required β-reduction, implicated order of processing, thereby violating the Church-Rosser theorem, etc.»

91. Thus, for example, in this variant the normal-form of the following λ-expression would be the identity function λx.x.

92. One can of course imagine more complex (i.e., non-functional) syntax as well, but semantics is the present issue. A similar technique could also be applied to the grammar, making the issue of whether any expression or even sub-expression is *grammatically* a holistic function of the text as a whole. (There have been computer languages which, as it is said, "cannot be parsed except at runtime," and which therefore cannot

types σ (σ_1, σ_2, etc.) assembled in according with grammatical rules of the form $\sigma_i \rightarrow \sigma_{io}(\sigma_{i1}, \sigma_{i2}, \ldots \sigma_{ik_i})$, with appropriate equivalences established among the various σ_{ij} so as to support any desired recursion. By assumption, we take θ to be a theory of the interpretation of texts as a whole. And finally, to be dramatic, suppose we include the "escape" rule discussed above: σ → `eject`(σ'), for arbitrary types σ and σ'. Then a compositional semantics for Σ, which we might call C_Σ, can be defined as follows:

1) *whole texts*: $[\![\tau]\!]\theta$

2) *complexes:* $[\![\sigma_{io}(\sigma_{i1}, \sigma_{i2}, \ldots \sigma_{ik_i})]\!] =$
$\lambda x \,.\, ([\![\sigma_{io}]\!]\ (\lambda y_o.([\![\sigma_{i1}]\!]\ (\lambda y_1.([\![\sigma_{i2}]\!]\ (\lambda y_2 \,.\, \ldots$
$([\![\sigma_{ik_i}]\!]\ (\lambda y_k \,.\, x(\ulcorner y_o(y_1, y_2, \ldots y_{ik_i})\urcorner)))\ldots)))))$

3) *escape:* $[\![\texttt{eject}(\sigma)]\!] = \lambda x \,.\, [\![\sigma]\!]\theta$

4) *atoms:* $[\![\rho]\!] = \rho$

As is evident, C_Σ is essentially vacuous. All it does is to step through the text, emitting the value of the `eject` operator if there is one,[93] and otherwise passing the entire text, *uninterpreted and as a whole*, to the holistic textual interpretation function θ (which, for all this theory cares, might as well be one particular individual's feminist analysis of political insinuations applicable only to Emily Dickenson's poems).

What's the point? There are four.

First, it is essential to recognize that, once a principle of *direct correspondence* is set aside, mapping terms (nouns and noun phrases) in the language under investigation onto the objects in the task domain that they intuitively "refer to," etc., then

be compiled, such as for example TECO, the language in which the still-popular EMACS editor was first written, as well as the first versions of Smalltalk.)

93. Strictly, it emits the value of the first `eject` construction that does not contain another `eject` construction within its argument.

nothing can be concluded about the nature of a semantical theory from the mere fact that it is compositionally specified. And note that direct correspondence is violated even by the classical strategy of introducing a sense/reference distinction, or one between intension and extension, and constructing a semantics in which terms are in the first instance assigned "meanings" (senses or intensions) rather than "referents." It is not only, once direct correspondence is set aside, that one can no longer conclude that the theory is compositional in any intuitive sense, or that it is genuinely committed to anything like a correspondence-theoretic semantical approach; it does not even follow that the semantical realm in terms of which θ interprets texts need be in any way classical (formed of classical ontology).[94]

Second, what the example illustrates is that, as long as a modernist framework is sufficiently powerful, then if one is willing to set aside some of the framework's default but informal and hence usually unenforced and often not explicitly articulated presuppositions, it often turns out that one can "code up" or "implement" any theory one wants on top of the framework—thereby deftly sidestepping its modernist assumptions to any extent that one pleases. This illustrates at extraordinarily important point that often arises in contexts where computing and semantics are both at issue: if language $\mathcal{L}_1$ has semantics Σ_1, and language $\mathcal{L}_2$ semantics Σ_2, and Σ_2 is in some way incompatible with Σ_1, then although one cannot *translate* $\mathcal{L}_1$ into $\mathcal{L}_2$ in a meaning-preserving way (literally: cannot translate expressions σ_{1i} of $\mathcal{L}_1$ into expressions σ_{2j} of $\mathcal{L}_2$ with the same meaning), it is often nevertheless possible to

94. I am not saying that analyses of the semantics of natural language framed in terms of sense and reference are not intuitively correspondence-theoretic. What it is, "intuitively," to be a correspondence theory is not that well defined. All I am claiming is that whether the semantics meets this (informal) criterion cannot be deduced merely from the fact that the semantical rules are framed in compositional form, as for example they are in $\mathcal{C}\Sigma$.

implement $\mathcal{L}_1$ in $\mathcal{L}_2$ (or in a purely descriptive case, *describe* $\mathcal{L}_1$ in $\mathcal{L}_2$), thereby ducking the restrictions of $\mathcal{L}_2$ so as to allow one to preserve Σ_1.[95]

Third, the example illustrates a feature that is generally characteristic of such "encoding" approaches. It is not just that the semantical machinery[96] is typically not used, in such practices, to map linguistic ingredients onto simple objects in what would intuitively count as the domain of interpretation; what it *is* used to map them onto is often a realm of functions that take as arguments simple or abstract structures built *out of the expressions themselves*. Note how similar this this practice is to the use of term models in logic, a practice on which prudence suggests I not comment here.[97] Suffice it to say that programming language semantics employs exactly this kind of coding trick to give mathematical expression to very substantial forms of non-locality, non-functionality (computational analogues of what logicians would think of as non-transparent reference), and the like.

---

95. There is nothing uniquely computational about the case. A declarative or logical analogue would be to describe $\mathcal{L}_1$ in $\mathcal{L}_2$, rather than translating expressions in the former in the latter. Linguistic ascent, quotation, etc., are typically more marked, in declarative contexts, however, than implementation is marked in computational ones. On the contrary, promiscuous implementation is virtually ubiquitous in computational practice, making anything other than purely behavioural semantics theoretical challenging.

96. In this example, the so-called semantical brackets '⟦' and '⟧' … «???»

97. One comment I cannot resist. As one might predict, given the discussion in the text, I in general view the use of term models as being as much of a "cheat" as the content-free compositional theory laid out above.

The late Jon Barwise, mentioned earlier (a preëminent logician, editor of the *Handbook of Mathematical Logic*, and co-author, with John Etchemendy, of the popular logic textbook *Language, Proof and Logic*) used to say that, if anyone claimed to have shown that any logic whatsoever, other than an essentially trivial one, was *complete*, then they had almost certainly cheated—probably by using term models.

Fourth and finally, to draw out an issue related to previous sections, it should also be apparent how handily the pressures and predilections of blanket mechanism are served by deploying modernist machinery in this way. The semantic commitments of the modernist framework are cleverly ducked; all that the modernist machinery is used for is to "implement" complex behavioural patterns. *Step by step, advertently or inadvertently, semantical issues, or at least semantical vocabulary and machinery, are drawn deeper and deeper into the clutches of engulfing mechanism* (§5).

This is meant to be an introduction, not a detailed analysis. Many more things could be said: the considerations adduced above address the issue of correspondence-theoretic semantical analyses, for example; they do nothing to seriously broach the topic of classical ontology and presumptions of intrinsic identity. But I will leave that and other issues to the papers that follow, to the introductions to the individual sections, and to future work.

For now, I will close with just one more observation—a third reason pertaining to the apparent lack of influence of the discursive critique on the theoretical and practical frameworks underlying contemporary infrastructure. While I have made no systematic investigation, and am certainly in possession of no hard evidence, it is my sense that contemporary programmers are far more aware than external observers might guess of fundamental problems or at least inadequacies in currently available modernist techno-theoretical frameworks. In terms of philosophical predilection, in fact, their sympathies may lie largely or even wholly with the discursive critique. But that, at least so far, is where the matter ends—because of the fact that that is what is widespread: a *critique*. What is lacking is an *alternative*: a positive, technically-developed framework, capable of underlying the construction, from bottom to top, of systems with millions and billions of parts.

The grip of the modernist machinery will not be loosened, in sum, by more critique, or by opening up communication channels between the humanities and the sciences and engineering—by requiring programmers to obtain degrees in discursive traditions, for example, or even take courses in science and technology studies. Rather, what we need to do is to develop viable alternatives that, recognizing the sting of the critique, *show us how to build*. Moreover, those alternatives must be explicitly formulated (pace, or rather embracing, the alternative understanding of what explicit formulation comes to). So long as they stay discursive, and retain the form of critique, I do not believe that the discursive traditions can do much more than they already have. In their own terms, in fact, one could declare the critiques a success; that is why mounting yet another critical analysis will seem, to some readers, perverse. What we need is a sufficiently detailed analysis to open up the possibility of construction. At present, in fact, if I had to summarize the current situation, I would say that what practice has done—what practice is doing—is: (i) to implement complex systems, whose behaviour and structures are more in line with phenomenological and poststructural sensibilities than they are with classical modernism; but (ii) to implement them within classical, modernist machinery, since that is all that is on offer. This situation contributes to the rather alchemical state in which we find ourselves.

And so we stumble forward—stuck in a bit of a quandary. To leap forward, to unleash radical progress, we need to forge alternative foundations. These pages by no means accomplish that lofty goal—though helping to move us along in that direction has certainly been their constant motivation. Perhaps they can at least serve as a point of reference on what developing such an alternative will entail.