Introduction

"I see!" said a voice at the back of the hall, breaking the silence. "You are trying to articulate, in language that only philosophers can understand, intuitions that only computer scientists can have."

Still haunting, thirty years later. But it is time for change.

1 Project

The aim of this essay is to provide a comprehensive, systematic, philosophically rigorous analysis of the foundations of computing. Or anyway that is how things start. Surprisingly, given computing's vaunted status, and in spite of more than fifty years of mathematical theory—to say nothing of unrelenting hype—these foundations remain remarkably murky. The situation has recently begun to improve, with a scattering of papers, volumes, and special issues on the philosophy of computing and information.[1] Nevertheless, many standard terms of art—*program*, *process*, *algorithm*, *symbol*, *data structure*, *information*, *implementation*, *architecture*, *complexity*, *state*, *nondeterminism*, *digital*, *virtual*, *online*, and the like—remain only partially excavated, bereft of satisfactory analysis. There is not even agreement on the most basic questions: what computation is, whether it is a concrete (physical) or abstract notion, etc.[2]

1. See Floridi (1999, 2004, 2008), Adriaans et al. (2008), and several special issues of *Minds and Machines*: "Computation and the Natural World" (2009 v. **19**:4); "Dynamic Emergence and Computation" (2008 v. **18**:4), "The Philosophy of Computer Science" (2007 v. **17**:2), "Hypercomputation" (2002 v. **12**:4), "Effective Procedures" (2002 v. **12**:2), and "What is Computation?" (1994 v. **4**:4).

2. Notions *per se* are by and large viewed as abstract. By an *abstract* or *con-*

The depth of naïveté may not be immediately evident. Technological progress races forward, computer science proceeds apace, computational metaphors permeate public discourse, and—especially tellingly—terms of computational art work their way into widespread theoretical explication, without much of anyone noticing conceptual difficulties. In fact for many, such characterizations have become so natural as to have subsided from visibility, in spite of their being relatively recent in the overall scheme of things. It is certainly no longer taken to be odd to construe biological organisms as "information-processing systems," to identify "bugs" in organizational procedures, or to characterize learning to play the piano as "compiling" knowledge of how to play chords into muscular memory—even if such descriptions were incomprehensible as recently as a generation ago. Similarly, within a span of a few decades, researchers in biology, linguistics, anthropology, physics, art, and other fields have grown content to rest explanations on what are (or have come to be treated as) essentially computational notions—*complexity*, *explicitness*, *information-carrying*, etc.—unproblematically considering such moves as serious and substantive commitments.

Yet relying on computational ideas is a little like skating on thin ice: all goes well so long as you keep moving. Stop and probe, and cracks open up—the gloss of solidity is lost. Hang around a while, to figure out what has happened, and one finds oneself plunged into the metaphysical deep. Who can say, these days, what is real and what is virtual, what is genuine article and what mere simulation? Are online phenomena abstract or concrete,[3] formal or informal, symbolic or non-symbolic, real or representational? No one seems to know, for sure.

...

crete notion, here and elsewhere, I mean a notion that applies to or holds of something abstract or concrete, respectively. By these lights, a paradigmatic Platonic realist would label the concept *chair* concrete; *number*, abstract.

3. Of course computers—the devices one hooks up to the internet, buys online, etc.—are concrete objects. What is at issue is whether the property of *being computational*, exemplified by these devices, rests on what are ultimately abstract or concrete metaphysical footings.

Putting things this way is standard—but buys into a distinction between computers and computations that I neither want to endorse in advance, nor to agree with later on. See footnote 18.

The cracks in the surface reflect no superficial disarray—something that could be healed with more education, better communication…or just a bit more time. Even the appearance of consensus is illusory. The more fundamental the questions, the more agreement unravels. It is unclear, for example, whether such elemental properties as *being computational*, *carrying information*, *being an algorithm*, etc. are: intrinsic, like mass and momentum; abstract, like the number four; relational, like being well-loved; or requiring of external ascription or interpretation, like the meanings of books and text.[4] The notion of *complexity*, similarly, increasingly thought to lie at the foundations of computer science, biology, and physics,[5] opens up into profoundly problematic issues, unresolved by any current framework—mathematical, philosophical, or other stripe. What is complex from one point of view (a stupefyingly intricate arrangement of organic molecules, say) may be simple from another (a single rose), and complex, but in an entirely different way, from a third (a suicide gesture). Similarly for computational examples: how can it take 28 million lines of code to implement 17 functions in a simple black-and-white copier? How can one study complexity without studying the *ways* in which something is or is not complex? And what kind of science would be a study of *ways*?

Methodologically, it is not even evident whether computation or computing—especially in full real-world dress, shaping and shaped by the world's rough and tumble—will succumb to scientific analysis at all, or whether the phenomenon is an essentially human practice, making it more amenable to study from the social sciences, humanities—maybe even the arts.

It is stunning that these questions have received so little attention. Perhaps it is because of computing's engineering focus—and the

4. Or anyway as mass, momentum, numbers, being loved, and meaning are *thought* to be. The status of such predicates or properties, objects or abstractions (to span a few classical binarisms) will itself turn out to be vulnerable to the analysis to come.

5. There are even those who believe complexity is destined to undergird a whole new eponymous science. See Wolfram (2002), Kauffman (1993), Simon (1962), and the foundational papers in complexity science listed on the web site of the Santa Fe Institute at (retrieved March 30, 2010):
www.santafe.edu/library/foundational-papers-complexity-science/

madcap pace of technological development. Perhaps it is because the experience on which that development is based is sufficiently communal—sufficiently shared, intuitively, by practicioners "in the know"—to have remained largely tacit. Perhaps it is because computer science was spawned from other disciplines (logic and metamathematics, primarily, in the first half of the twentieth century), suggesting that we can inherit computing's foundations from its intellectual forebears. Or maybe it is simply because computing, even in these opening decades of the new millennium, remains relatively new. Whatever the reason, the central notions of one of society's most widely heralded developments have remained remarkably conceptually untheorised.

The problem is not so much that basic questions have been entirely overlooked. Rather, it is that the foundational discussions that have taken place have been conducted almost wholly within the discipline.[6] As a result, most of the debates have been specific: focusing on the adequacy of particular theoretical frameworks, in order to explicate or resolve debates among alternatives, rather than digging deeper to explicate the fundamental character of the concepts in terms of which such proposals are framed—or, more broadly, on which the whole field rests. Extant discussions are in this way more analogous to the debates that took place between classical and quantum physics at the beginning of the twentieth century, or to contemporary ones about integrating quantum mechanics and cosmology, than they are to the rise of a disenchanting mechanist philosophy several centuries earlier—even if the wildfire spread of computational notions and metaphors throughout the academy, and throughout society more generally, are more akin to those early modern developments than they are to twentieth-century scientific reconfiguration.[7]

---

6. Except for some conducted within cognitive science and the philosophy of mind; see footnote 12, below.

7. Some (perhaps even many) readers will come to this investigation assuming that the rise of computation falls within—perhaps even represents a triumph of—a philosophy of mechanism. That this is not so will be one of the main conclusions I will argue in what follows. This is one of several conclusions that will substantially "up the ante" on the significance of computing.

Illustrating this internalist character, a number of theorists have recently criticised some widely accepted models of computing—especially the historically canonical ones based on mathematical models of abstract automata (including Turing machines).[8] These formal models, some writers argue, are intrinsically incapable of explaining the vast, messy, distributed, real-time networks and systems of interactive agency that not only increasingly structure our lives, but are also starting to infect the metaphors and assumptions that permeate wider intellectual explanation. And so, the critics continue, we should reject—or at least downplay—the historically central notions of discrete automata and "stored program computers" interacting (at best) with passive tapes and memories, and instead take up a suite of more lively alternatives: asynchronous interacting agents, dynamic networks, non-representational robots—even models based on the theatre.

I am sympathetic to many of these internal critiques. Turing machine and von Neumann architectures will come in for their share of criticism here as well (though what matters about them will also need to be distilled and preserved). But though proposals for conceptual rearrangement can highlight foundational difficulties, by themselves they do not automatically clarify conceptual footings. For one thing, the new proposals need conceptual assessment, too. What is it, exactly, to be an *agent*? Does anything that acts make the grade? Is terra firma an agent, when it gravitationally "acts" to bring us back to earth? And what, to open up a philosophical minefield, is *representation*, such that non-representational systems do not do it? Does representation require explicit *symbols*—or do systems of tacit meaning count? For that matter, what is it to say that meaning is *tacit*? Do migratory birds tacitly represent the movements of the stars? Does the earth's orbit tacitly represent the location of the sun? And what does *interaction* mean—does it require energy to be exchanged? Or information? And how do energy and information relate, anyway? Suppose you promise to call before noon if you are not coming to dinner—and I stand, silently, listening to the clock strike…and slowly start to smile. Have we interacted?

8. E.g., Stein (1996), Wegner (1997), Wegner and Goldin (2003), Goldin and Wegner (2008), and Milner (1989).

The importance of clarifying the foundations of computing is made more urgent by the fact that computational notions have been so widely exported to both adjacent and far-flung fields. As already noted, all sorts of other disciplines—logic, mathematics, physics, linguistics, economics, psychology, biology, neuroscience, anthropology, even literature and art—are recasting central questions and fundamental claims in computational or informational terms. It has been a long time since computing remained within the unique purview of computer science departments. In fact it is rare, these days, for each substantial department or division within a major university not to offer its own course dealing with information and computation, at least with the impact of computing and information processing, within that discipline's context. But if all these "customers" do not have (or are not given) a rigorous understanding of the concepts they are adopting, how are they—or anyone else—to understand their reformulations? Even if increasingly populated by the "digitally-born",[9] it is too much to assume that the members of these disciplines will share the entire fabric of tacit understanding that underwrites their fields' internal critiques and debates—and hence that they will be able to understand, at any more than a superficial level, the internal critiques and debates themselves.

Take cognitive science, a preëminent employer of computational metaphors, which will come in for its share of scrutiny here.[10] On reflection, it turns out that many of the most passionate debates in that field, over the last fifty years, have turned as much on disagreements about what it is to be a *computer* as on views about what it is to be a *mind*. Consider Searle's infamous "Chinese room," in which he argues that purely syntactic (or formal) symbol manipulation cannot be constitutive of mind. Or the neo-Heideggerian argu-

9. The phrase 'digitally born' is usually used of works: documents, art pieces, etc. But it can be usefully applied to people, too—i.e., to those who grew up with computers, and so take them pretty much for granted. Tending (as suggested earlier) to be in their thirties or younger, and to have lived in industrialized nations, these are those for whom digital technologies are a default (e.g., in cameras)—and the predicate 'digital' consequently almost passé.

10. Whether the mind is (metaphorically) *like* a computer, or whether the mind (literally) *is* a computer, is a fundamental question in artificial intelligence and cognitive science; I gloss the difference here only in order to set the issue aside till later. See chapter 2 of Volume 1.

ment in Dreyfus' seminal *What Computers Can't Do*, that ineffable backgrounds and tacit expertise underwrite all genuinely human capacities. Or dynamicists' call to reject computational models in favour of those based on dynamical systems theory. Or those who argue against computationalism in favour of neuroscience or biology.[11] These critics all rest their case on conceptions of computing that are not only far from obvious, but, I will eventually argue, are demonstrably (in fact outright empirically) false. As arguments that the mind is not computational, therefore, they all fail—not because they do not rest on genuine insights into the mental (i.e., not because the mind is not the way that they claim), but because they misconstrue or underestimate what it is for something to be computational.

Since details are instructive, consider in particular the wave of "anti-computational" sentiment that coursed through the ranks of cognitive science in the 1990s. Based in part on the failures of artificial intelligence (AI) to live up to its triumphalist predictions of the 1960s and 1970s, cognitive science's originating "computational theory of the mind" was scathingly attacked from all sides, in favour of a broad collection of alleged alternatives: *dynamical systems*, *cognitive neuroscience*, *artificial life*, *complex adaptive systems*, *evolutionary epistemology*, *behavioural robotics*, and so on. Time and time again, these new proposals were defended in articles formulated in such terms as "dynamics *vs*. computation."[12] The problem, independent of the merits of the new proposals, was that it remained unclear what exactly it was about *computing* that was being so roundly rejected—or, more seriously, when the conception of computing was relatively clear, whether it was *actually true of computation*. If a model of computing that is subjected to critique or derision as a model of mind turns out not to hold of real-world computing systems, either—such as Unix, Microsoft Word, or the iPhone—the sting of the attack is considerably lessened, if not outright nullified.

For better or worse, the anti-computational arguments of that final decade of the twentieth century were not isolated events.

11. See for example Searle (1980, and 1992 ch. 9), Dreyfus (1979), van Gelder (1995, 1998), Port and van Gelder (1995), and Edelman (1992).
12. Cf. van Gelder (1995): "What might cognition be, if *not* computation" (emphasis added).

They led to a major period of upheaval, which eventually resulted in something of an overall intellectual sea change in the field of cognitive science. John Haugeland had labeled the traditional model of mind on which cognitive science had been founded "GOFAI," for "Good Old Fashioned Artificial Intelligence."[13] It viewed intelligence as a relatively *abstract*, *discrete*, *serial*, *explicitly symbolic*, *rule*-governed form of *individual rationality*. Based on the logical tradition from which computation itself had originated, this founding model was paradigmatically exemplified by detached, autonomous, formal systems: systems that proved theorems, for example, were programmed to play chess, or solved mathematical puzzles. The rather disembodied and a-contextual classical model is now roundly rejected by a majority of cognitive scientists, at least as a model of everyday living and coping with events, in favour of what is often called a "situated" approach. The situated approach instead views its subject matter (cognition, consciousness, mind, personhood, the human condition, etc.—there is some debate as to how to characterise it) as *physically embodied*, *embedded* in a hosting environment, at least *non-symbolic* and perhaps even entirely *non-representational*, intrinsically *active*, fundamentally *social*, and highly *context-dependent*. Concrete real-world navigation and contextually dependent improvisation, rather than abstract ratiocination, is taken, from the new perspective, to be the paradigmatic form of intelligent activity.[14]

What does all this have to do with computing? Just this: most cognitive scientists view this sea change as a rejection of the computational model of mind. What is odd, though—to say nothing of tremendously telling—is that computation itself, in the same twenty years, underwent (and continues to undergo, up through the present) *much the same paradigm shift*. Throughout this same period, though in completely separate literatures and communities, computer scientists pressed (and, again, continue to press) for more concrete, contextually-located, embodied and embedded forms of

13. The term 'GOFAI' is introduced in Haugeland (1985, ch. 3, p. 112), and contrasted with 'NFAI'—"new-fangled AI"—in Haugeland (1997, ch. 1).
14. Smith (2001); available at (retrieved March 30, 2010):
cognet.mit.edu/library/erefs/mitecs/smithbc.html

both practice and theory. It looked for all the world, and continues to look, as if computing itself—the real-world, concrete, live phenomenon, if not (yet?) the deliverances of mathematical theory—is undergoing the very same sea change as its cognitive cognate. Computational practice, in other words, is running rough-shod over all the same classical views of what computing is, including—especially including—the "logicist" conception presumed by classical cognitive scientists. The actual computational systems that underlie modern real-time embedded operating systems, for example, such as those that power the packet routers that relay internet traffic around the world, to say nothing of those that control the brakes in virtually all contemporary automobiles, are as "concrete" and "situated" as anything that cognitive scientists have ever recommended as constitutive of mind.

These developments are of foundational interest, as opposed to merely reflecting various forms of speciation and development within the imaginaries of computer science and the public, because they raise into question whether there is (or ever was) anything *intrinsically* abstract, discrete, symbolic, rationalistic, etc., about computation itself, as presumed in what computer science still calls "the" theory of computation. Is computation *necessarily* as it is characterised in what is unquestioningly accepted to be its foundational theory, in other words, and as is increasingly scorned by dismissive cognitive scientists? Or was that rationalist, formal, autonomous conception itself merely one particular, historically situated *species* of computing, or *idea* about computing,[15] perhaps appropriate in its day, maybe useful for some purposes—perhaps even (who knows?) not ultimately useful at all, an early fledging stab at a theory, destined to be relegated to the dustbins of history?[16]

15. It would be merely an idea about computing, not a species of computing, not just in case extant computers were not one hundred percent correctly described by it (since all conceptions and/or ideas about computing are idealizations), but rather in case those extant machines or systems did not function, *as computers*, in virtue of exhibiting or manifesting those properties or characteristics that the idea espoused. To delineate such a difference requires a foundational understanding of what it is to be a computer—which is to say, a theory of computing of the sort being sought here.

16. The automata-theoretic conception continues to underwrite reigning mathematical computability theorems. So much the worse for them, as we will see.

Breathtakingly, no one knows. Our understanding of computing is barely deep enough to allow us to ask such questions—let alone to come up with a satisfying answer.

The issues can be put more generally. What in our current understanding of computing, conceptually unchallenged as it has remained, is authentic, genuine, or necessary? And what is gratuitous, and/or historically particular? What powers and what limitations stem from each aspect? What, empirically, is true of real-world practice, and what is false? How much of current theory is not in fact essential; and what that is constitutive of practice remains untheorised? Without a more adequate understanding of conceptual foundations—without a deeper understanding of the phenomenon of computation itself—such questions are impossible to answer.

The goal of this investigation is to map this uncharted conceptual territory. The project can be understood, in part, by analogy. Foundational inquiry has great historical precedent in science, meta-science, and the philosophy of science. For a century physics has spawned intense discussions about the proper interpretation of quantum mechanics, the nature of the measurement problem, the collapse of the wave function, the role of the observer. Logicians, similarly, have debated the epistemological impact of Gödel's celebrated incompleteness theorems, the nature of formality, and the status of intuitionism, constructivism, and Platonism. Biologists discuss units of selection, question the impact of emergence on prospects for naturalistic reduction, and characterize the gene as an information carrier.[17] Many sciences, as we plunge forward into the new millennium, are concerned about the impact of uncertainty, non-linear complexity, and turbulence on questions of determinism and predictability. And of course there are analogous debates outside science—for example in anthropological, cultural, and literary discussions about the (inter) relations among subjectivity, objectivity, and material participation, about the nature and politics of participant-observers, about the reflexive character of social and cultural analysis, and myriad other issues.

In spite of their unparalleled importance, however, foundational

17. A notion of information, of course, that, if probed, is widely presumed to rest on computational foundations.

questions in computing and computation[18] have yet to receive the same kind of systematic scrutiny. The situation is betrayed in the current makeup of philosophy departments. No one can deny that the intellectual exploitation of notions of computing, information, architecture, digitality, etc., are equal to (if they do not vastly outstrip) the theoretical impacts of relativity or quantum mechanics. Yet whereas the philosophy of mathematics and philosophy of physics are ubiquitously recognized as mainstay specialisations within philosophy of science, itself a central area within philosophy, and whereas philosophy of biology has recently undergone explosive growth around the world, joining (if not supplanting) math and physics as a première philosophy of science, courses and specializations in the "philosophy of computing" remain virtually non-existent.[19] Historically, in the 1930s and 1940s, such philosophy of computing as did exist took place in the traditions from which computing was spawned: logic and metamathematics (cf. Gödel, Kleene, Church, Gandy, and others).[20] In the latter decades of the twentieth century, philosophical discussion about the nature of computation was largely conducted by philosophers of *mind* (Dennett, Dretske, Dreyfus, Fodor, Haugeland, Searle, van Gelder, etc.)—i.e., from writers whose expertise rested on intuitions and scholarship about perception, cognition, and consciousness, not from experience writing operating systems and Java applets.[21]

18. It is traditional to distinguish 'computation' and 'computer'—taking a computer to be a concrete physical mechanism that "implements," "realises," or "executes" a computation, with the last in turn taken to be abstract. The term 'computing' is more general, and is informally used to denote active processing. How these words should be used, however, is something that I take to be central to the forthcoming analysis, rather—crucially—than something to be decided in advance. In particular, and especially in advance, I do not want to embrace the abstract/concrete distinction implicit in the normal usage of 'computation'/'computer.' So for the time being I will use the three essentially interchangeably.

19. As I write this, there are only a very small handful of philosophers, worldwide (notably including Jack Copeland and Luciano Floridi), beyond a handful of recent students of my own, who cites "philosophy of computing" on their curriculum vitae.

20. See Gödel (1992), Kleene (1936), Church (1936), Turing (1936), and Gandy (1980, 1988).

21. John Haugeland deserves recognition as an exception; with a partial back-

As stated at the outset, it is time for this to change. As attested by the centrality of computationalism in (at least classical) cognitive science, by the growing use of 'information'-based explanation in biology, and by other uses of computational explanations and metaphors across the academy, fundamental discussions throughout the intellectual pantheon are increasingly resting on notions that stem from some admixture of computer science and concrete computational practice. Historically, it may have been enough to treat computation as an intellectual stepchild, as if it could travel on another discipline's passport—be taken as a form of mathematics, as an idea about cognition, as a branch of engineering. But given the growing body of technical results, the sophistication of computational technology, and—most importantly for foundational analysis—the perfusion of computational ideas throughout surrounding intellectual life, the consequences are vast enough for computing to warrant its own, independent, foundational inquiry.

So that is the goal: to provide a comprehensive, systematic, philosophically rigorous analysis of the conceptual foundations of the most important technological development of the last hundred years.

Or anyway, as I say, that is how the project begins.

2 Computation

The investigation will consist of two phases.

The first phase will be broadly analytic. Its aim is to deconstruct the framework of assumptions and concepts on which our current theoretical understanding of computing rests: notions of *formal symbol manipulation*, *effective computability*, *Turing machines*, *finite state automata*, *information processing*, *digitality* and *digital state machines*, "*rule following*," Newell and Simon's notion of a *physical symbol system*, and the like. These notions are not normally viewed

...

ground as an engineer, he constructed his own printer (in the 1960s) by attaching solenoid-driven paper-clips to an electric typewriter; and programmed Postscript throughout his academic career. Fred Dretske similarly started life as an engineer—a fact likely relevant to his "If You Can't Make One, You Don't Know How it Works" (Dretske 2000), reminiscent of Feynman's famous "What I cannot create, I do not understand," found on his blackboard at the time of his death in 1988 (Hawking 2001, p. 83).

as "internal alternatives." That is: they are not conceived as different architectures, suitable for different purposes or problems, or as competitors for how computation should be founded or theorized. Rather, tied together in various ways, the cluster as a whole is taken to found or constitute or prop up the reigning "mythos" about what computation is.[22]

En route, a number of additional notions—*program*, *process*, *procedure*, *data structure*, *algorithm*, *computability*, *complexity*, etc.—will also be examined, partly on their own, but more substantively in terms of these prior fundamental notions. Towards the end of the first phase a variety of more recent architectural suggestions will also be investigated, including those of *self-organising systems*, *connectionism*, *artificial life*, and *generalised dynamics*. In contradistinction to the first set, these last ideas have been proposed in more of an alternativist than foundationalist spirit (even if they are sometimes presented as if they were contenders for some sort of computational crown). Perhaps because of this, they tend to rely, in their fundamental characterisation, on the initial ones—e.g., on notions of what is "computationally possible," what effective computing comes to, etc., notions that are taken in turn to rest on the original roughly Turing-machine-theoretic consensus. It is the initial list, therefore—*formal symbol manipulation*, *effective computability*, *Turing machines*, *information processing*, and *digitality*—that will bear the brunt of the inquiry, at least in the first phase.

One of the very first results of the analysis will be to show that, even if these ideas are thought to be equivalent or compatible, they are in fact much more different—conceptually, historically, sociologically, extensionally, etc.—than is normally recognized. For

22. Computing is boring, some readers will say, especially younger ones—no more "mythic" than electricity, or blood. There is something right in that, even if blood and electricity both had their time in the sun. But it would be a mistake to allow computing's stunning passage from mythic to mundane to obscure either: (i) the imaginative discontinuity it effected, just a few short decades ago, in our conception of ourselves, the world, and our place in it—even if some of that was hype, and much else, as I will argue, mistaken; or (ii) the deeper impact it has in fact had, and will continue to have, on our cultural and intellectual understanding.

analytic purposes, therefore, I will label them "**construals**"[23,24] of computing—and in a series of in-depth analyses trace the origins and take the measure of each one.

The analysis of each construal will consist of two parts. The first will be to ask what intuitions it is based on: why it is believed, what is right about it, and what of value it can contribute to a more adequate, successor account. I take this positive reconstruction—a sustained attempt to secure what is deep and insightful and true about each of these extant ideas—to be one of the project's most important long-term contributions. At the same time, however, and second, I will also document their failure: their ultimate inability to meet straightforward conceptual, explanatory, and empirical criteria of adequacy. For to cut to the bottom line, I will ultimately argue that although each construal rests on a deep insight, no one of them, nor any group in combination, is strong enough to stand up the jury to which I will ultimately defer: *doing justice to concrete empirical practice*. There will be time later to examine the rationale behind and merits of this standard; for now it is enough simply to mark it with a term. Throughout, candidate proposals will be held accountable to what, with deference to Hutchins,[25] I will call **computation in the wild**: that eruptive body of practices, networks, techniques, systems, and behavior that has so palpably revolutionized late twentieth and early twenty-first century life.

3 Analysis

Analyses will be conducted at a pace of roughly one construal per volume. They will start in Volume II, with the construal having the deepest historical and logical roots: that of *formal symbol manipulation*. Volume III will turn to the Turing-theoretic conception of *effective computability*, which, although often thought to rest on the

23. In their first occurrence, words to be used technically in these volumes will be presented in bold face, as here. As they are introduced, all such terms will be collected and given short descriptions in an evolving accompanying glossary, available at:

www.ageofsignificance.org/aos/glossary

24. 'Construal' rather than 'model' or 'architecture' in order to emphasize the fact that each of these ideas, as well as their combination, has been proposed in order to get at (or to help get at) *what is fundamental or essential or in common to computing as a whole*.

25. Hutchins (1996).

logical conception, in point of fact departs much more radically from its forebears, I will argue, than any literature has recognized. Similarly, subsequent volumes will sequentially take up *information processing*, *digitality* and *digital state machines*, etc.

Before this series of iterated analyses can begin, however, some preliminary groundwork is required, in order to ensure that the analysis is conducted as rigorously, constructively, and non-presumptively as possible—and in an adequately reflexive critical spirit, given that presupposition is of course ineliminable. That will be the task of Volume 1: to establish an intellectual "base camp" from which specific assaults on each construal can then reasonably be launched.

The aim of the present Introduction is contrapuntal to that of Volume 1: it aims at overview, not preparation. I want to make some preliminary comments, respectively, on: (i) the framework and approach that makes the entire project possible; (ii) some of the positive results and reconstructions to which it will ultimately lead; (iii) concomitantly, a few of the substantive problems it will uncover; and (iv) how, in response, the plot has to change—and deepen—midstream.

For change it will. We, too, will be thrown into the metaphysical deep, but not without preparation. Out of the detailed reconstructions and considerations uncovered in the first analytic phase of the investigation a second metaphysically constructive phase will ultimately develop.

3a Meaning and mechanism

At the most fundamental level, it will turn out that computing is best understood as a **dialectical interplay of meaning and mechanism**. That is not to say we can start out by *presuming* that computers are both mechanical and meaningful; to do that would be to preëmpt the investigation in ways that, as already noted, I will struggle as much as possible to avoid. Rather, the claim is merely that it will turn out to prove most intellectually fruitful, in the long haul, to examine computation from those two intersecting perspectives.

Why *dialectical*—as opposed simply to interplay? This question cuts deep into the analysis. At this stage I intend no commitment

to Hegelian synthesis, Marxist materialism, or any other technical connotation. I simply want to mark a result that will come out of the investigation: that, though not simply opposed, neither will the notions of *meaning* and *mechanism* prove comfortable confederates. They will relentlessly wrench and tear at each other, each being constantly reconfigured by the other—in something one might characterize as productive instability, if instability were to one's liking. Enough, here, to say that the two notions deserve pride of place in establishing an investigative frame, if for no other reason that their juxtaposition will test the very mettle of the investigation.

With respect to *meaning*, it will be natural to many, and only contentious to some (though those "some," interestingly enough, will include wide swaths of computer science), to view computers as interpretable, symbolic, representational, information-carrying, or in some other way *meaningful devices*: systems whose ultimate nature derives not merely from causes and effects, from the bumpings and shovings they inflict upon the world and that are inflicted upon them, but from the fact that those causes and effects, those bumpings and shovings, *signify* this or that situation, *carry information about* this or that state of affairs, *represent* this or that phenomenon, *mean* this or that thing—or are at least *able to* mean such things, perhaps for the system itself, though understanding what that means and what it would take to achieve will be a long time coming, but at a minimum are *able to be interpreted as representational or meaningful or content-bearing* by or for their users, external observers, or other interpreting systems.[26] In philosophical jargon this would be summarized by saying that computers, in one way or

26. If, as I will ultimately argue, computers are in fact taken to be meaningful or meaning-bearing in some constitutive sense (rather than just being examined through such an investigative frame), then what it is to be meaningful is either something that a theory of computing should explain, or else something that should be derived from an "external" theory of meaning—i.e., of semantics or (as philosophers say) intentionality. At this early stage I do not want to take a stand on which of these options is to be preferred, or on any substantive issue about what meaning is—e.g., on whether "to mean" is an intrinsic, or extrinsic, or observer-relative, property, and a spate of other issues. Developing computationally appropriate answers to such questions should be the goal, not the premise, of an investigation of this sort.

another, are **intentional** systems or entities. In due course we will need to take on intentionality in this sense as a substantial subject matter in its own right; for the time being I will use 'meaningful,' 'meaning-bearing,' and 'intentional' (as well, later, as 'significant') as rough synonyms.

On the *mechanism* side of the dialectic, it will again be natural to most readers to recognize computers as subject to substantial and fundamental constraints of *effectiveness* or *mechanical potency*. At an intuitive level, though it will be only a moment before I mention some challenges to this view, one can take 'effective' as meaning something like causally (i.e., physically) effective, or at least causally or physically implementable or realizable—i.e., to take computers, as concrete entities, to be things that, in virtue of being computers, can do real work in the material world. This, I take it, is to view computing as unlike what those who believe in such things take divine rumination to be: an abstract phenomenon unconstrained by considerations of finitude, efficacy, or the limits of embodiment. On the contrary, there is an almost Leninesque resonance to what is perhaps the most widely agreed tenet underlying the entire computational field: that computation has to do with *what can be done*.

In this respect, computers would seem to epitomize the so-called "mechanist philosophy" that has gripped human imagination since the Scientific Revolution of the 16th and 17th centuries, and the subsequent development of the "natural sciences." In fact it is not too much to say that after three or four hundred years of relentless scientific expansion, and especially in light of the domestication by computational models of such apparently sacred human topics as consciousness, altruism, sociality, and the mind, that for many people the computer has come to represent the zenith of a mechanist world view.

We will see about that later. For now, and much more modestly, it is enough to adopt, as an opening gambit, a view of computing as involving this dialectical interplay of meaning and mechanism. Moreover, what starts out as investigative framework will systematically develop into substantive thesis—i.e., will move from hypothesis to claim.[27] Computers do indeed, I will argue, and necessarily,

27. "How could it not," a critic will argue, "since you started out with it?" Well, I did not start out with it originally.

involve both mechanical and meaningful aspects.[28] *Sans* constraints of meaning or meaningfulness (i.e., some flavour of intentionality), computers would amount to nothing more than "machines"—or even, as I will ultimately argue, to "stuff": mere lumps of clay. Unless it recognizes meaningfulness as essential, even the most highly perfected theory of computation would devolve into neither more nor less than a generalized theory of the physical world. *Sans* some notion of efficacy or mechanism, conversely, no limits could be either discerned or imposed on what could be computed, evacuating the notion of constraint, and hence of intellectual substance. Freed from all strictures of efficacy or mechanism, from any requirement to sustain physical realizability, computation would become fantastic (or perhaps theistic): meaning spinning frictionlessly in the void.[29]

3b Potency

While it may seem the soul of reasonableness to analyze computing in terms of a meaning/mechanism dialectic, I need to deal right away with a conception that in some people's eyes may seem to challenge this approach, and to challenge the intuition mentioned a moment ago, about computation having to do with causing things in the real world. I refer in particular to a conception of effectiveness that underwrites the currently accepted mathematical theory of computation and computability, and that derives in turn from the logicist background out of which Turing's conception of computing arose.

It is common, in this tradition, though by no means universal, to take the Turing-machine theoretic construction and the associated computability and/or recursion theory to rest on an *abstract* conception of efficacy or effectiveness, unrelated to any considerations or limitations stemming from physical or material realiz-

28. Again, and as usual: computers, computing, computation, the lot. See footnote 18, as well as the following several paragraphs.

29. The phrase 'frictionless spinning in a void' is from McDowell (1996), who uses it in reference to the idea of unconstrained spontaneity (e.g., pp. 11, 18, 42, 50, 66, 67).

ability.[30] The approach is most famously associated with a remark attributed to Edsger Dijkstra: that "Computer science is no more about computers than astronomy is about telescopes."[31] The idea (though whether this is taken to be an interpretation or result of the theory, or the motivation behind it, who knows?) is that one can (i) develop or demonstrate an abstract notion of computational efficacy or effectiveness on purely mathematical or logical footings; and then (ii) define, in its terms, a mathematically pure but nevertheless constrained and thereby substantial notion of computation,

30. Two informal generalizations may be helpful in understanding who is likely to sympathize with such a view. In my experience, younger computer scientists are likely to view computational efficacy as physical or concrete in origin, with those of an older generation more disposed to see it as mathematical or abstract. In addition, those originally trained in computer science, computer engineering, and/or computational practice (i.e., programming) are more liable—again, in my experience—to lean towards a concrete or physicalist interpretation of the computability constraints than those trained in logic or mathematics, with the latter tending to view the constitutive notion of computational efficacy not only as abstractly modeled but as (ab)originally or ontically abstract.

An additional note, in passing. Much later—well into Phase II—I will argue that the issue of how mathematical theories are or should be interpreted, which at the moment is widely viewed to be "extra-theoretical," outside the scope of theory proper, will, in virtue of the metaphysical account to which (I will argue) computation leads us, become a "theory-internal" affair. If the account that I ultimately defend has merit, that is, it will no longer be possible for two people who subscribe to the same mathematical theory to have different "interpretations" of it, since the issue of interpretation will by that time have been brought inside the theory's theoretical scope. (Yes, for better or worse, mathematics is going to be at stake.)

31. While widely quoted and ubiquitously attributed to Dijkstra, the saying is listed as "unsourced" at: http://en.wikiquote.org/wiki/Edsger_W._Dijkstra

I have always found Dijkstra's comment stunningly off-key—not only because of its commitment to a fundamental difference between computational and causal forms of efficacy, but also because of its apparent confusion or conflation of subject matter and tool. As regards the former issue, the subproject within this work of studying and resolving differences, ambiguities, and commonalities between and among potency predicates is an attempt to make good on my long-standing sense that something has gone horribly wrong in a tradition that can valorize such a claim. The latter issue—about the relations among theory, tool, and subject matter—will come to the fore when we analyze Turing's original conception of a computing machine (in Volume III).

untainted by any contingencies and particularities of the concrete physical world.[32]

Put it this way. By **potency predicates** I will mean all those predicates or properties[33] that people have informally taken to have to do, one way or another, with "getting things done." Thus I will take the potency predicates to include such notions as causal powers, computational effectiveness, perhaps materiality and physicality, etc. The abstract or mathematical interpretation of computability theory can thus be viewed as a claim of ontological or even logical independence between two notions[34] or *kinds* of potency: (i) *efficacy* or *effectiveness*, on the one hand, of a sort relevant to computation, and (ii) *materiality* or *physicality*, on the other, applicable to all concrete things. If God created the physical world on Monday, an adherent of such an independence view would think, then he set up the constraints on computational efficacy (e.g., established Gödel incompleteness and Turing non-computability) on a different day—later in the week, or the night before, or whatever, but anyway in such a manner that he could have changed either, even radically, without that making any difference to the other.

I will eventually argue that this approach fails, contrary to the bulk of current theory, but the suggestion is worthy of consideration, especially given the historical weight resting on its shoulders. My aim in these few paragraphs is not to assess it, however, but

32. Some may object to the idea that the laws of physics are contingent, but if our investigation is to range widely and deeply, then so should our imaginations.

33. I continue, with malice aforethought, to run roughshod over distinctions between map and territory (i.e., between predicates and properties).

34. Throughout these introductory pages (and well into the first several volumes), I will play fast and loose with the difference between properties, general or universal phenomena, and types (such as causality or efficacy),on the one hand, and *notions* or *concepts* of those properties, phenomena, and types, on the other. The reason stems in part from the constructive character of the metaphysical account to which the investigation will ultimately lead, in part from a desire to avoid pedantry this early in the story, and in part from a sense that excessive clarity at this stage would require fronting one metaphysical stance (realism, constructivism, formalism, idealism, etc.) over another, thereby eroding the sympathy and participation of whole classes of potential readers.

only to reassure readers that I will be able to give it adequate assessment and defense within an investigative framework formulated in terms of a meaning/mechanism dialectic. In particular, I will treat theories that take computation to be abstract not as approaches that *abandon* constraints of mechanism, but as approaches that *take the notion of mechanism to be abstract*.[35]

The example illustrates a general point. Each time a new construal is taken up for investigation, I will want to ask about its potency predicates—i.e., about what notion of "being mechanical" or "being effective" it relies on: what it says about the abidingly necessary source of that constraint, whether it views that source of constraint as abstract or concrete, necessary or contingent, whether its take on computing would be the same or different in a world whose physical laws were radically different from ours, etc.—as well as inquiring about which interpretation of its claim, in answer to all these questions, does the best justice to computational practice. Since for the moment I am merely proposing the meaning/mechanism dialectic as a framework for analytic investigation, not (yet) adopting it as a substantive thesis, I can afford to remain open as to what each construal takes 'meaning' and 'mechanism' to come to.

In sum, there are many reasons to take the "meaning/mechanism" characterisation of computing as primary—i.e., to use it as a top-level structural principle, around which to organize the inquiry. It has deep historical roots; so long as we are open-minded about what meaning and mechanism are taken to come to, it will give us intellectual access to all standing computational construals; and, in spite of its ultimate dialectical pungency, in the long haul it will prove as profound a conception of computing as any we will encounter. In what follows, therefore—methodologically at first, substantively in due course—I will assume that computers involve (are constituted by, exemplify, have to do with—it is hard, in advance of theory, to know just what word to use) some kind of dialectical interplay of these two (yes, mythic) notions.

35. Understanding how physicality, materiality, and concreteness relate will be one of the goals of Phase II.

4 Positive results

Before turning to foundational problems that will be unearthed through application of this framework, it is useful to consider a few of the inquiry's positive, computation-specific results. Among other things, this will allow us to see some of the reformulations that will be required in what we already partially understand.

4a Effective Computability

One of the most consequential results, foreshadowed above, will be an argument that we need to recast our understanding of the mathematical theory of computation and computability in concrete terms. As also already mentioned, this theory is widely known by the name "*the* theory of computation," though as I will suggest in a moment, that is not a label it deserves.

Fundamentally, as I have already said, it is widely agreed that the theory of effective computability focuses on "what can be done by a mechanism." But three conceptual issues have clouded its proper appreciation—even its formulation.

First, as suggested above, and in spite of its use of such words as 'effectiveness' and 'mechanism,' the theory is almost always characterised abstractly, as if it were a branch of mathematics. While *per se* this may not necessarily be a problem, the approach leaves open to question the relation between the allegedly abstract notion of efficacy and very real constraints imposed by physical law, physical reality, hanging in the air, without explication or theoretical legitimacy. It is not enough to assume that God made two independent creative acts; their independence should be defended, if not explained.

Second, the theory is generally imagined to be a theory defined over numbers, or over functions on numbers (numbers, sometimes, used to model other Turing machines). Why this should be problematic will require explication, but one concern arises straightaway: the nature of numbers (what *they* are) is hardly a topic on which science, society or the academy has developed crystalline consensual clarity. To rest a foundational theory of computing on numbers is therefore to inherit metaphysical ambiguity from one subject to another (rather than, as in the case of physics, merely using numbers and other mathematical entities to *classify* a sub-

ject matter, which is presumably less ontologically fraught, even if epistemically less than transparent).

Third, with respect to conceptual interpretation, and in consort with our analytic approach, the theory of Turing machines is generally given a semantical cast: the marks on the tape of a Turing machine are taken to be *representations*—representations in general, or a delimited species of representations called encodings, again, of numbers, functions, and/or other Turing machines.[36] Yet the resulting theory does nothing to explain the representational or encoding aspects on which it apparently necessarily relies.

In almost exact contrast to the received view, I argue to the following three conclusions.

4a · i Effectiveness as concrete

First, in spite of being open to a catholic variety of intuitions "on entry," as it were, as already suggested I will ultimately claim, once they are subjected to rigorous analysis, that all attempts to derive an abstract notion of effective computability fail, and necessarily so; that, far from being abstract, the efficacy that constrains and thereby gives substance to computing is fundamentally and ineliminably *concrete*: a direct consequence of the physical nature of patches of the world (specifically, in its traditional formulation, a consequence of "digitized" patches of a classical physical world). In reality, that is, or so anyway I will claim, the theory of computability is no more "mathematical" in underlying character—i.e., in terms of the underlying metaphysical regularity to which it is giving theoretical voice—than any other theory in physics. As a result, there is no reason it should be framed any more mathematically than other physical theory (even if, as I will explain in a moment, it uses mathematical structures to model that physical reality).

36. Other formulations dispense with "machines" entirely, including all mention of tapes and marks, and formulate the notions of computation and "computable" purely in terms of arithmetic operations on the integers. Although these conceptions are not explicitly semantical, they are even more abstractly characterised than in the standard semantical—i.e., representational—model. As I will endeavour to show (needless to say, this is a large goal, and is the sort of aim that makes the project substantial) these two characteristics—of being semantically and being abstractly characterised—are two sides of the same coin, representing the same unitary failure.

4a · ii Numbers and encoding

Second—and far more contentiously, though I believe it can be convincingly demonstrated—I will argue not only that computability and recursion theory derive the requisite efficacy constraints from physical laws, but that they do not deal with meaning or intentional issues at all. That is: the superficial appearance of a semantical aspect to the interpretation of our recursive and computability-theoretic analyses is illusory.

It would certainly appear that classical computation deals with semantically laden issues—or specifically, in the case of Turing machines, with the *manipulation of representationally interpreted configurations of marks*. From a theoretical point of view, in fact, Turing machines are analysed as comprising a two-level structure, isomorphic to the way in which logic is analysed when studying consequence, entailment, soundness, completeness, etc. In both cases, concrete structures at one level (marks on Turing machine tapes, in the computational case; formulae or expressions, in logic) are taken to denote mathematical or semantic entities, at another level (numbers and functions, in the case of Turing machines; facts, models, and again numbers and functions, etc., in logic). The operations of the Turing machine controller are similarly treated as parallel to the formally-characterised steps of a logical inference system. And so on.

This view is explicit in the very first sentence of Turing's classic paper, in which computing was introduced: "The 'computable' numbers may be described briefly as the real numbers whose *expressions as a decimal* are calculable by finite means."[37] While Turing's statement might stand as a (stipulative?) definition of computable numbers, however, I argue that the view universally taken to follow from it—that computing should be analysed as structurally parallel to logic—is a simple but profound mistake. That is: the idea that, for purposes of developing a proper theory of computing, the marks on Turing machine tapes should be interpreted as representations or encodings of numbers, I claim to be false.

In its place, I defend the following thesis: that the true "encoding" relation defined over Turing machine tapes runs the other way

37. Turing (1936, p. 230).

around, from numbers to marks. Rather than its being the case, as is universally assumed, that configurations of marks encode numbers, that is, what is actually going on is that *numbers and functions defined over them serve as models of configurations of marks, and of transformations among them*. It is this structure that makes the situation analogous to the rest of the physical sciences, where numbers are used to model velocities, masses, energies, etc.—except that in the computational case, for reasons to be explained, what is modeled is not described in terms of evident *units*. Rather than an (allegedly binary) mark such as '11011' encoding the number fifty-five, for example, as traditionally assumed, what is really going on, in terms of warranting the theoretical conclusions that are ultimately drawn, is that the mark '11011,' and other mark configurations effectively isomorphic to it, are modeled, in the theory, by the number fifty-five—roughly in the way that the escape velocity from earth's orbit, in miles per hour, is modeled with a number near 24,000.[38]

The truth, in sum, is 180° off what we have all been led to believe. (It is also 180° off from what the people who formulated recursion theory thought, but so much the worse for them.[39])

38. 24,000 mph ≈ 11.2 kilometers/sec, a rough estimate of vertical escape velocity at the earth's surface. Rockets aiming to escape earth's orbit typically depart the earth's surface much more slowly than this, partly because they continue to accelerate while airborne, but also since they need achieve only that escape velocity holding of the trajectory they occupy at the point at which they cease propulsion—which, since it will be higher than the earth's surface, requires a lesser escape velocity. Escape velocity also depends on the location where the object departs (since gravity is not uniform world-round), which direction it aims (in order to exploit or counter the earth's rotation), etc.

39. I will have more to say later on about how the intentions of the authors of various theoretical frameworks, including the intentions of the developers of recursion, computability, and complexity theory, bear on how we should understand the results of their theorising. In the present case, I will attempt to show, via the computational theory of mind, not only why it is that the computability constraints necessarily derive from physical constraints, but also, and ironically, via a reflexive application of the computational theory of mind, why someone would think that computability constraints were mathematical in nature, even if in fact they are not. (It is not always necessary to explain why it is that people who hold false beliefs have come to be in the epistemic state in which they find themselves; but as ammunition in setting aside their conclusions it does not hurt.)

Moreover, to make matters worse, rather than the relation between marks and numbers being a semantic relation within the genuine subject matter of computability theory, I claim that its actual home is "one level up," as part of the theoreticians mathematical modeling equipment. That is: whereas the mark-to-number relation was assumed, on the classical account, (i) to be a semantic one, and (ii) to be part of the genuine subject matter of the theory, on my reconstruction the modeling relation from numbers to marks (i) exists at the level of the theoretician themself[40] (i.e., at the level of the theory qua theory, alongside other theoretical equipment), and therefore (ii) is not part of the genuine subject matter after all. Rather, the numbers and functions universally associated with marks and machines are *used* by theorists to model configurations of marks, to put it in philosophical jargon; they are not *mentioned*, as part of the subject matter, at all.[41]

Ironically, that is, Dijkstra got it backwards. It is the numbers and functions that are the telescope; the computers are the stars.

It follows that computability and complexity results (such as

40. As in Smith (1996), I use 'they' and 'them' as syntactically plural but semantically singular third-person personal pronouns of unmarked sex. Though the usage is common, perhaps even to the point of banality, it is striking that reflexive pronouns take their number semantically rather than syntactically, as in such standard constructions as "Do yourself a favour." Principle thus commits me to uses of 'themself,' even if the word (currently?) seems awkward.

41. For philosophers and logicians: The terms 'use' and 'mention' are normally taken to be applied only to words, expressions, formulae, etc., with an expression being *used* when it occurs (paradigmatically) in a referentially transparent context, so that its contribution to the semantic value of the whole in which it exists as a part derives from facts about what it *names*, *represents*, *describes* or *denotes* (i.e., from its semantic value), whereas an expression is *mentioned* when (paradigmatically) it occurs within quotation marks or in some other way is that *object* or *semantic value* of another. Thus the term 'Boston' is used in the sentence "Boston is the capital of Massachusetts," but mentioned in the sentence "The word 'Boston' contains either 5 or 6 letters, depending on whether one counts tokens or types." In this work, I will (conservatively) extend normal usage to say that the object or semantic value of a term that is used is thereby mentioned, so that it becomes correct to say that the largest American city east of New York is *mentioned* (twice) in "Boston is the capital of Massachusetts."

that very large integers can almost surely not be factored into their prime factors in polynomial time on a deterministic machine) will all have to be reformulated, in terms of what can and cannot happen to concrete arrangements of physical devices. Similarly, all results that involve "deciding," "figuring out," etc.—such as the halting theorem, in which it is proved that no machine can "decide" whether an arbitrary other machine will halt on an arbitrary input—will also need reformulation, since the claim that a machine is "deciding" is a conception that depends on interpreting the marks on its output tape ("deciding" is an intentional, not mechanical, notion). More strongly, nothing whatsoever follows directly, from any result in recursion theory or the mathematical theory of effective computability, about what can and cannot be *decided* about anything—since deciding is an intrinsically semantical activity.

4a · iii A theory of marks

In this way we are inexorably led to the third point: the very strong conclusion that computability and complexity theory—i.e., what goes by the name "the theory of computation"—fails not because it makes false claims about computation, but because *it is not a theory of computation at all* (wild or tame).[42] Instead, it is something like a mathematical (i.e., mathematically modeled) theory of *differential (effective) causality*: a theory about what configurations or arrangements of concrete physical stuff can be switched or moved around or transformed, by finite physical processes, into other configurations or arrangements of perfectly concrete stuff, in what sorts of amounts of time, using what sorts of amounts of space. That is all. It is a theory of effective mechanism, pure and simple.

42. Note that this claim—that the so-called theory of computation fails as a theory of computation because it does not deal with computation's intentionality—is a result that should be agreed even by someone (e.g., Searle) who believes that computation's intentionality is inherently derivative. I myself do not believe (and will argue against the view) that computation's intentionality is inherently derivative, but even those who think it is derivative must still admit that it is an intentional phenomenon of some sort. For *derivative* does not mean *fake* or *absent*. If "derivatively intentional" is not taken to be a substantive constraint, then we are owed (e.g., by Searle) an account of what *does* characterise computation. Does he think that computers are undistinguished material mechanisms?

In particular, it is not a theory of *meaningful* mechanism, in any sense beyond the vapid one that, qua mechanisms, meaningful mechanisms must obey mechanical laws. If, that is, as I will claim, computation does indeed involve a dialectical interplay of meaning and mechanism, this reconstituted computability theory—which we should now call a (mathematical) theory of efficacy, or a (mathematical) theory of differential causality—will *apply* to computers, just as it applies to people, galaxy formations, the condensation of gases, etc., but it is not a theory of *computation*, in the sense that it is not a theory of computers or of computing *as computational*.

4b Formal symbol manipulation

A third conclusion, of a somewhat different sort, will come out of the analysis of the formal symbol manipulation construal of computing. This theory, which directly underlies the development of formal logic, deals explicitly—and genuinely, in my view—with the manipulation of *symbols*, and is therefore unlike the mathematical theory of computability just mentioned. That is: it focuses on what we have just seen that the officially received "theory of computing" ignores: the still-crucial semantic aspect of computing. Because it takes both meaning and mechanism seriously, the formal symbol manipulation construal stands a better chance, at the outset, of doing justice to computing than the theory of effective computability, with which it is often allied or even confused.

Nevertheless, the formal symbol manipulation construal will also be shown to fail[43]—not, this time, because it fails to be about computing, which on the contrary it genuinely is, but rather because it is *false*. Real-world computing—what, as I have said, I call "computation in the wild"—turns out *not to be formal*, on any discernible reading of that recalcitrant and contentious predicate. In particular, the common assumption that computers involve, or are constituted by, the manipulation of symbols "*independent of their semantics*" proves to be too strong, empirically. To reduce a complex investigation to a single sentence, I can put it this way: the formal-symbol manipulation construal ultimately fails because, in many real-world cases, semantic domains are constitutively implicated in

43. See Volume II.

the computational processes defined in, over, around, and through them. Real-world computers are *actively involved* in their subject matters, that is; they are consequential players in the very worlds they represent. In their comings and goings, they make essential effective use of, and essentially affect, the very states of affairs that their symbol structures are (semantically) about. The concomitant "co-constitution" of symbols and referents, in the assembly of a computational process, defeats the sorts of symbol-reference independence that formality presumes.

Rather than consisting of an internal world of symbols separated from an external realm of referents, in particular, as imagined in the formal symbol manipulation construal (and as exemplified in paradigmatically formal systems, such as axiom systems about abstract numbers, or symbol systems about far-away and independent task domains, such as NASA databases about planetary precession), real-world computational processes are often what I will call **participatory**: they involve complex paths of causal interaction between and among symbols and referents, both internal and external, cross-coupled in complex configurations. That is not to say that *semantical properties are causally efficacious*; that is a stronger result, which I will ultimately deny. But the participatory involvement is strong enough to defeat any attempt to define such processes as *formal*.

A note, in passing. The claim that semantics is not effective even though computation is not formal—i.e., that computation proceeds neither *independently* of the semantical properties of its ingredient symbols, nor *in* (causal) *virtue* of them—is an example of the sort of positive fine-grained insight we will be able to extract from reigning views that, overall, we have to discard. It is partly in order to win such victories that, as will be explained in Volume I, but has already been betrayed in the description of construals, I will adopt an investigative method far more essentialist in flavour than any substantive conclusion I will ultimately endorse.

4c Universality and equivalence

A fourth result has to do with the metrics of equivalence universally used to show that one machine, or machine type, is "equivalent" to another, as well as to undergird the notion of a "universal"

computer (since to be universal is to be equivalent to any other). What I will argue—to be blunt—is that this equivalence metric is a terrible idea: tremendously misleading, the historical progenitor of fifty years of intellectual misadventure, and a continuing distraction to our ability to understand the powers and limitations of meaningful mechanisms.

One immediate problem with the ubiquitously-assumed metric is that it is almost vapidly coarse-grained. So-called "effectively equivalent" computers can differ by more than most of the major disputes in science—e.g., those in current cognitive science and philosophy of mind (e.g., on whether minds are representational, dynamic, and/or functional; whether they can be adequately characterised behaviourally, or whether a proper and satisfying intellectual treatment must advert to internal mechanism; etc.). Claims that universal computers can do "anything" must also be taken with a very large grain of salt, since tapping out a rhumba, making coffee, and sending email are outside their ken—"out of court," as it were, though why computing's court (or ken) should be restricted in these ways is by no means clear in advance. Certainly that restriction, whatever it is, should at most be taken as a "post-theoretic" conclusion, not pre-theoretic assumption.

But the difficulties go even deeper. Some fundamental problems follow from, or are anyway pointed towards by, the reformulation of computability theory as concrete, mentioned above. But they will also depend on questioning yet another almost never challenged assumption: that computational processors ("computers" "CPUs,"[44] Turing machine controllers, etc.) are "solid"—concrete, physical, etc.—in contrast to programs, configurations of marks on the tape, etc., which, on the contrary, are normally deemed to be transient, ephemeral, or abstract. Once it is appropriately recognised that programs are concrete arrangements, too, and the impact of that recognition is reflected back into the foundations of the theory, the sharp conceptual divide between processor and program begins to erode.

It is perhaps worth saying, in passing, that challenging the program/processor divide is in some ways commonsensical. There has

44. "Central processing units," though the acronym has more currency these days than what it stands for.

always been something very odd in how one (standardly) proves that one machine—call it m_1—is equivalent to another machine, m_2. Strictly speaking, one does not show that m_1 *by itself* is equivalent to m_2. Rather, one shows that a pair consisting of m_1 *plus a program* p is equivalent to m_2—where on any ordinary notion of complexity, theoretical importance, etc., the vast majority of the burden is borne by p! Even more strangely, in proving a machine m_1 *universal*—i.e., equivalent to any of an infinite number of different m_k—one uses different pairs, with a different program p_k for each one. And then one says—in what must surely be one of the oddest "results" ever to have been intellectually promulgated—that m_1, *by itself*, is equivalent to *all* of the different m_ks!

Why is the entire variegated set of programs p_k so thoroughly ignored, when they (severally) consist of the bulk of the complexity and substance, and all the credit given to (vastly simpler) m_1? So far as I have been able to determine, the answer is not a happy one. Consider the following analog. Suppose I wish to prove that the 99¢ pen in my hand (call it q) has universal novel-writing power—that it is capable of producing any writable novel. For any given novel n_i, I show that q can produce it, by pairing q with the novel's author a_i, who then proceeds to write the novel down. A ludicrous result, of course; in each case the credit belongs to the *novelist*, not to the *pen*. Moreover, not only is the pen essentially irrelevant; *nothing* substantial has been shown *universal*, since I paired the pen with *different* novelists, in order to span the set of possible novels.

So why, in the $m+p$ combination, do we give credit to the typically rather simple machine m, rather than to the usually vastly more complex program p, whereas in the $q+a$ combination, it is ludicrous to give any credit to pen q? And not only that; since there are different programs p for each machine to which m is proved equivalent, why do we have any confidence in the idea that one *single*, stable thing is genuinely equivalent to a *highly disparate group* of other things?

My own belief is that the reason stems, ultimately, from two metaphysical facts or intuitions—each far from explicated. The first has to do with the machine m's being active, energetic, something that *does* things, in contrast to program p, which is usually (taken to be) passive, something to which other entities respond. Second,

the reason may stem from our differentiating things or devices, on the one hand, from their possible arrangements or configurations, on the other. Neither distinction is one I would want to go to court to defend. Devices, for example, in their own ways, are simply arrangements of chemicals—and so a device/configuration distinction, especially at the level of foundational theory, is probably not something on which we should rest great confidence. In fact our willingness to embrace this entire form of argument—and, therefore, to take seriously any claim about computational equivalence (of a sort which is very common)—should, I believe, on reflection, start to erode.

Needless to say, this argument will not go down swimmingly, for all theoretical palates. Readers will undoubtedly have their own responses to the putative analogy. And it is not, of course—to set the record straight—that I am immune to recognising either (i) that one can implement just about any computational process on a general purpose computer, or (ii) that the ability to program something is a wondrous and extraordinary thing. On the contrary, I admit that both are astonishing and very powerful facts. But the question about programmability is one that will recur many times: *wherein lies its power?* What, really, is going on? And how, theoretically, should we give it its proper explanation?

Above, I said that all results in the theoretical body of work known, today, as the "mathematical theory of computation," or "mathematical theory of computability," or "mathematical theory of complexity," must ultimately be reformulated in concrete terms. My ultimately strategy—which picks up on the intuition that the difference between a machine and a program may involve issues of animation and energy, of *activity* vs. *passivity*—is to frame a concrete reconstruction of the famous universal equivalence proofs in terms of what I call a **motor theorem**, with roughly the following content: given a motor m_1, and an adequate stock of other passive parts,[45] I can assemble a configuration p of those parts, such that the resulting device, consisting of motor m_1 appropriately connected up

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

to a purpose-built configuration of those parts *p*—a device of potentially Rube Goldberg complexity—will produce isomorphic behaviour (again, "equivalent" according to a non-trivial isomorphism metric) to that of any other machine that you can build.

Is this motor theorem impressive? That is: should we be impressed that such a theorem can be proved? Who knows? Personally, I should not have thought so. Most active concrete devices consist of some number of motors, gears, pulleys, containers, pipes, etc. It does not seem particularly odd, at least to me, that with just one motor, of sufficient size, plus an indefinitely large supply of other *perfect*, *friction-free* devices—switches, ropes, pulleys, etc., something on the order of the world's most amazing Meccano or Erector set—one could construct a functionally isomorphic device to any given device, so long as the metric of equivalence one is mandated to meet is sufficiently broad, as it is in this case.[46]

But perhaps normative assessment should wait until we can work through the reconstruction in detail. At any rate, something like this, I will claim, is what an adequate conceptual understanding of the notion of a universal computer comes to, on proper reflection.

And so on.

As I say, these few results are just the beginning: they are meant to give some indication of the sorts of argument and conclusion to be reached in the full investigation. Slowly and steadily—through issues of *formality*, *computability*, *information*, *discreteness* (*digitality*), etc.—our understanding of the computational realm will have to be systematically dismantled, en route to being rebuilt.

5 Problems

These results alone still do not comprehend the full scope of the conclusions to be reached. For as well as all these specific adjust-

46. One thing that the idea of a vast Meccano set does not highlight is the idea that configurations of parts might be viewable as *instructions*—as intentional specifications or prescriptions, not merely as assemblies or mereological parts. Could the insight about programmability lie in the fact, not that functional devices of arbitrary complexity can be made in a world in which perfect friction-free parts are legion, but in the fact that, in such worlds, unbounded complexity facilitates the construction of *interpretable* assemblies?

ments, reformulations, and new results, there are three even more fundamental problems, which will ultimately not so much overshadow what comes before them, as reconfigure both their formulation and their significance.

5a Meaning

First, as suggested above, no theory or construal in the standard list, including any reformulation of a received theory, is powerful enough to comprehend the semantic or, as philosophers say, *intentional*, dimension of computing: issues of what symbols, representation, information, interpretable processes, etc., *mean*. Given our investigative stance of viewing computation as involving both "meaning" and "mechanism," this can be put very simply: no extant theory, nor any obvious incremental revision of any standard theory, adequately comprehends "meaning."[47]

Two facts about meaning, in my view, are especially poorly understood. The first is its "long-distance" or relational character: how symbols, representations, data structures, thoughts, information, can "leap over" gaps in time, space, and possibility, so as to orient us (or computers, or their users) to long-ago, far-away, even fictional states of affairs—e.g., when thinking about the temperature on the far side of the moon, or the myth of Sisyphus. Ultimately, as we will see, the long-distance, relational character of meaningful structures is somewhat defined in terms of—though in opposition to—issues of computational effectiveness. One of the most profound facts about the semantic or intentional character of computational

47. A natural question will arise for philosophers: of whether various proposals about semantic content that have been formulated in the philosophical literature, such as teleo-semantic or biosemantics ideas that what an expression means has to do with the role it has played in subserving its proper function, or the "indicator semantics" idea, originally developed by Dretske, that something means that with which it is counterfactually correlated, can be appropriated and used in the computational realm. Those are substantial questions, to which I am not ready here to post a simple reply—except perhaps to say this much: that it is clear neither (i) that evolutionary considerations will apply to arbitrary computational mechanisms, nor (ii) that the indicator-semantics view, which (as is widely recognised) requires a distinction between normal and background conditions, is compatible with the strong metaphysical conclusions to be discussed in the next section.

processes is the way in which they facilitate reference to situations that stand beyond their "effective grasp"[48]—where the notion of "efficacy" on which that claim rests is exactly that aspect of mechanisms explained by the theory of effective computability, appropriately reformulated in concrete terms. If a system—creature, network router, robot, mind—cannot "reach out and touch" some situation in which it is interested, another strategy, deucedly clever, is available: it can instead exploit meaningful or representational structures in place of the situation itself, so as to allow it to behave appropriately with respect to that distal, currently inaccessible, state of affairs.[48] This is the home of representation; analysing it in computational settings will provide us with a usefully non-anthropomorphic setting in which to investigate its fundamental character.[49]

The second poorly-understood aspect of meaning, even more difficult, has to do with its *normative* character: the fact that truth, reference, meaning, and other intentional properties involve issues of value, worth, virtue, significance. One of the positive results we will extract from the formal symbol manipulation construal of computing, in spite of the failures documented above, is that it gestures towards an understanding of computation as a system of *normatively-governed causal transitions*. For this to be a substantive reconstruction, however, and for us to understand it, we need to reach some understanding of normativity—no small task, and not something that will be given us by any mere simple reconstruction or local repair of the formal symbol manipulation construal.

5b Mechanism

So that is the first of the three major problems: we do not yet understand issues of meaning well enough to achieve our goal of a comprehensive theory of computation. The second, perhaps more surprising, but again foreshadowed in the specific results adumbrated above about the recasting of computability theory as a concrete theory of differential efficacy, is that we do not really under-

48. Haugeland (1998, Essay 4).

49. Note that this non-local, non-effective aspect of semantics is one of the things that fails to be appreciated by an undue focus on the program-process relation, of the sort typified by studies of programming language semantics.

stand mechanism, either—the other side of the dialectic. But the situations are not parallel.

As one would expect, given three centuries of natural or "mechanical" science, our understanding of mechanism is relatively advanced—far more so than our understanding of meaning. But a number of general concerns remain.

First, as will especially come to the fore when we discuss information, the notion of differential efficacy, which I am claiming to be the real subject mater of computability theory, is particularly germane with respect to the sort of long-distance correlation that underwrites the metaphysical fact that the world supports one state of affairs carrying information about another. But identifying what forms of local differential causal efficacy are relevant to long-distance correlation is not a trivial task (among other things, this is where the earlier example about signaling an intention to come to dinner is relevant: learning something by *the failure of something else to happen*).

Second, as indicated in the earlier comment about its not being expressed in terms of units, notions of "differential efficacy" (roughly: causally distinguishable states, like two states of a switch, or two discriminable pits on a piece of plastic, as on a DVD) need to be studied independent of particular considerations of mass, force, momentum, etc. While still perfectly concrete, understanding this form of differential efficacy involves understanding mechanisms at a more abstract level than is traditional in the (classical) mechanical sciences, but—crucially—more concrete than something purely abstract, such as mathematics, or even, as already argued, as they have been treated in traditional computability theory.[50] Formulating the right theory of mechanism for computational purposes, therefore, will involve delineating something of a "middle way," between the abstract and concrete, of a form not before recognised in science.

A third problem, which may on the surface seem minor or innocent, will prove of such magnitude that it will be one of the strongest forces driving the investigation into the full metaphysical reconstruction of metaphysics, ontology, and epistemology constituting Phase II. The issue has to do with the universal theoreti-

50. At least mathematics as normally—i.e., Platonically—conceived.

cal practice of identifying states (or state types) in terms of which to formulate analysis. Finite state machines are a staple of formal computational theory; when my companion silently signaled their interest in coming to dinner, they effected a change in my epistemic state; dynamical systems theory is framed in terms of system states; etc. The problem is that we have no theory as to what constitutes a legitimate state (or, again, state type). As illustrated by such examples as Putnam's Turing-complete rock and Searle's Wordstar-computing blank wall (to say nothing of Goodman's distinction between *grue* and *bleen*),[51] it is straightforward to individuate states in a sufficiently perverse way so as to obtain results that take complete leave of both common and intellectual sense. Silly examples, many readers may feel—a sentiment with which I have some sympathy. But no one has a theory of the difference between sensible and silly mechanical states. More seriously—and more pointedly—no one has a *mechanical* theory of what constitutes a legitimate mechanical state. More seriously yet, evidence will begin to mount that that the warrant for mechanical states *may not itself be mechanical*. And if it is not, what then?

A fourth concern is more meta-theoretic. Although it is true that three centuries of science have given us a remarkably powerful understanding of mechanisms of various types, we (somewhat reflexively) have not adequately *integrated that understanding*. What we have learned about mechanism is splattered across a variety of theories and sciences, in somewhat disorganized form. With respect to the abstract/concrete issue just mentioned, for example: (i) some sciences, including mathematical recursion theory and computational computability theory, as we have seen, treat mechanism as abstract, eerily free of any physical moorings; (ii) some, including physics, chemistry, and material science, take it to be concretely material, so that issues that deserve more abstract treatment (such as what type of "architecture" evolution stumbled on, when it discovered the uniquely powerful recursively combinatoric possibilities of carbon-based molecules) fail to be given adequate prominence; and (iii) some, including such engineering practices as software meth-

51. Putnam (1988), Searle (1992), Goodman (1955).

odology or circuit design, are tied into remarkably unreconstructed mundane material understandings of the world, as if such lay notions as *wire* were scientifically or naturalistically respectable.[52] In order to honour the promise implicit in the concrete reconfiguration of present-day theory alluded to above, these partial understandings will all have to be appropriately unified.

In sum, meaning is not our only problem. The mechanism side of the house is equally in need of rethinking.

5c Site not subject matter

The third difficulty with current theories of computing, beyond the fact that we understand neither meaning nor mechanism, is more surprising yet—and more consequential. I have already said that I do not believe we are in possession of an adequate theoretical understanding of computing. The problem, however is not just that we have not had an adequate theory of computing, in the past. Nor is it that we have no adequate theory, now. Much more strongly—and in what at first will seem a dismal conclusion—**we will never have such a theory, in the future**. We will never have a theory of computing because *there is nothing there to have a theory of*. Or rather, to put it more accurately: it will be a major conclusion of this investigation that neither computers, nor computing, nor computation, nor anything close by, are ultimately the sort of phenomenon that will sustain an intellectually satisfying, trenchant, powerful theoretical account.[53]

Computers, in the end, turn out to be rather like cars: objects of inestimable social, political, and economic significance, but not destined, per se, to be the subject matter of deep, generative, intellectually satisfying explanation. The reason is simple. Beyond manifesting a dialectical interplay of meaning and mechanism, computers *per se* do not have what I earlier said they must have, in order to constitute a genuine subject matter: they are not sufficiently *special*. As we saw, it is incumbent on a theory of computation to take 'computational' to be substantial property, for the class of computers to be thereby restricted. For there to be a "there, there"—for there to

52. I.e., in philosophical terms, were "naturalized."

53. As some philosophers would say, they are not a "natural kind"'—though for reasons that will eventually emerge I am unquiet about that notion.

be something for a theory of computation to be a theory of—the computational must constitute a determinate subset of everything that is or anyway could be the case.[54] But every candidate theory to be explored here, when subjected to rigorous examination, will turn out to fail: either because it is too *narrow*, in the sense of not applying to some things we ought to (in fact already do) call computers, or else because it is so *broad* as to be essentially vacuous, amounting to no more than a vapid form of physicalism.

In spite of the advance press, that is, I ultimately argue that computers prove, on inspection, not to be necessarily digital, not to be necessarily abstract, not to be necessarily formal—not necessarily to exemplify *any characteristic property making them a distinct species of the genus "meaningful material system."* Nor can the situation be rescued by Wittgensteinian resort to family resemblances, Roschian adversion to prototypes, Zadian reach for fuzzy sets, or feminist appeal to anti-essentialism. Perhaps ironically, one powerful consequence of positively reconstructing each construal, rather than just of demonstrating its failure, will be that, collectively, the reconstructions supply enough insight for us to realize that it is not a question of computing even being *vaguely* or *mostly* or *roughly* or *conjointly* as it has been characterised. On the contrary, we will arrive at a point from which the following is clear: computers are intentionally significant physical artifacts—the best we know how to build.

Period. There is nothing more to say.

6 Age of Significance

On first encounter, the conclusion that computing is not a subject matter will feel like defeat. Such a "result" might seem to incriminate the whole project, to eradicate the utility of the inquiry. (If there is no such thing as computing, why write—or read!—all

54. Note that this is true even if one were to argue that everything is computational. Physicalism holds that everything is physical (in some appropriate sense)—a thesis that many people believe. But it does not follow that there is nothing to being physical—that "anything goes," as regards physicality. On the contrary, it is exactly because to be physical *is* to be special, in ways that the laws of physics explain, that the doctrine of physicalism is substantial—has bite.

these volumes?) In fact, however, I defend almost exactly the opposite conclusion. That there will never be a theory of computing—and hence no computational theory of anything else, such as of mind—*is the most optimistic conclusion that anyone involved in the development of computation could ever have possibly hoped for*. It is a result that should warm the heart of even the most unregenerate computational triumphalist. It is not that computation is powerful enough to *take over* the world, as the hype-meisters would have it, or that computation will *change the world*, or even that computation will change our *understanding* of the world. All of these things will happen; but not because computing is special. If grand gesture is to the point, why not tell the truth: what computation is, *is* the world.

So that is the final conclusion of the first phase of analysis. Rather than being a delineable subject matter, warranting its own private or distinctive theory, worthy of being etched on the facade of the twenty-second century academic buildings, computing is instead a **site**: an historical occasion on which to see general, unrestricted issues of meaning and mechanism play out. And it is a site of immense importance—signaling the end of three hundred years of materialist natural science: science as a form of knowing restricted to the study of matter, materials, and mechanism, constitutively explicable in terms of causal regularities. In its place, computing and information technologies—together, increasingly, with biology, neuroscience, and nanotechnology, all of which are beginning to be understood in convergent intentional terms (such as in terms of information)—are serving as a midwife for an emerging era of intellectual history that will ultimately be as important, I believe, as the era of materialist science—an era I call the **age of significance**.

Imagine exploring an old house, and opening a door in a hallway, somewhere, expecting it to lead into an unknown room. To your amazement, it opens directly into an entire world—complete with fields, forests, panoramic vistas, cities and towns, people and projects, bustling activity. Your first inclination might be to think that you had tripped over a magical kingdom, complete unto itself—a virtual Narnia. But as you explore—take a few initial steps, then longer journeys, push on things and discover that they push back,

build houses, raise children, take journeys, and gradually explore and domesticate it—you discover that it is the same world you have always known. But not the "same old world," as it were, rendering entry through the magic door irrelevant. Rather, what the door provides is mysterious entrance into the same world you have always known, but with a much clearer, more energised, almost magically more powerful perspective.

Studying computing, in my experience—and perhaps even spending a lifetime marinating in computational systems—is something like that. I have gradually come to believe, over these last forty-two years, that it is something that those who build computational systems know, deep in their blood and bones, though they have had no way to say it. As suggested in that opening reaction to that comment from the back of the hall, which took place in the 1970s, a decade or so into this effort, I will make an initial attempt to say it here—but doing these intuitions more than cursory justice will take time, more time than I have, certainly, and in the end I suspect they are something that, even at best, can be only partially articulated, at least at this historical moment. For now, it is enough to say that whereas traditional natural science embraced causal explanations, in its efforts to account for matter, materials, and mechanism—physical stuff, in myriad forms—computing, I believe, is opening us, in ways we as yet barely understand, into an era of *intentional* sciences. Our intellectual scope is widening, coming to understand how the mechanical aspect of the world dialectically plays off against broader issues of meaning, interpretation, semantics, normativity, significance.

The particular role that past and present-day computational practice will play, in this qualitative shift—what history will ultimately recognise them to be, I believe—is an experimental, synthetic precursor to the emergence of this new theory: something we might call **semiotic alchemy**. Just as the 14th and 15th centuries were characterised by a wondrous wealth of disheveled, synthetic, untheorised material practices, subsequently shunned but now recognised to be essential precursors to the rise of science, so 20th and (at least early) 21st century computing, I believe, will be seen as an analogous spate of rag-tag, messy, untheorised practices—something that was initially conducted in cottages, basements, ga-

rages, and academic research laboratories, and that has now taken root throughout society—practices that embody a vast wealth of pragmatic and practical, though not yet very well understood, expertise.[55]

What these practices are leading us to—the realm they are opening up for us—is not a distinct corner of the world, or an isolated new phenomenon, or even a new understanding of a corner of the world or of a new phenomenon. Rather, as we will glimpse at the end of Volume 1, they are ushering us into a clearer, more trenchant, more potent understanding of the meaningful world as a whole.

I close with a final irony—one of many that will colour the trip. To indulge in glib caricature, I have said: (i) that computing is almost unutterably important; (ii) that there is tremendous amount of tacit understanding in the blood and bones of programmers; and (iii) in terms of explicit theory, that much of what we have said about computing, and have been told, is false.

How can a generation of programmers have come to embody such a deep implicit understanding of something so consequential if they start out being told falsehoods? I believe the answer has to do with the fact that, in the end, computation in the wild is more than anything else a *practice*. Though one might view the present investigation as an exercise in the philosophy of science, it might be more accurately located in the philosophy of engineering. I am not going to agree with Feynman or Dretske that we can understand only what we can build; as should be evident, I am not even convinced we understand what we *do* build. But with a trace of a Kantian smile, I would be the first to admit that, along with descriptive and theoretical, synthetic methods deserve a place in our understanding.

The practical character of computing has remained central in my imagination, during the lifetime that this project has been underway. Throughout the whole period, I have received countless communications from programmers and computer scientists—over coffee, in the midst of debugging sessions, in conferences, at bars and via email—hoping that it might be possible to *say* what they

55. Think of all those Silicon Valley hackers and entrepreneurs, strenuously labouring to turn web sites into gold.

feel they *know*, the absence of which, in some cases, has led to almost existential frustration. It is out of respect for this trust, and in sympathy with their intuition, though with no confidence that I have done their hopes justice, that I dedicate these volumes to them.

References

Adriaans, Pieter, Paul Thagard, John Woods, Dov M. Gabbay, and Johan Van Benthem, eds. 2008. *Philosophy of Information*. *Handbook of the Philosophy of Science*. North Holland.

Church, Alonzo. 1936. An unsolvable problem of elementary number theory. *American Journal of Mathematics* 58, no. 2 (April): 345-363.

Dretske, Fred I. 2000. If you can't make one, you don't know how it works. In *Perception, Knowledge, and Belief: Selected Essays*. Cambridge University Press.

Dreyfus, Hubert L. 1979. *What Computers Can't Do*. Harper & Row.

Edelman, Gerald. 1992. *Bright Air, Brilliant Fire: On the Matter of the Mind*. Basic Books.

Floridi, Luciano. 1999. *Philosophy and Computing: An Introduction*. Routledge.

Floridi, Luciano, ed. 2004. *The Blackwell Guide to the Philosophy of Computing and Information*. Blackwell.

Floridi, Luciano, ed. 2008. *Philosophy of Computing and Information: 5 Questions*. Automatic Press/VIP.

Gandy, Robin. 1980. Church's Thesis and Principles for Mechanisms. In *The Kleene Symposium*, ed. Jon Barwise, H. Jerome Keisler, and Kunen Kunen, 123-148. North-Holland.

———. 1988. The Confluence of Ideas in 1936. In *The Universal Turing Machine: A Half-Century Survey*, ed. Rolf Herken, 55-111. Oxford University Press.

van Gelder, Tim. 1995. What might cognition be, if not computation? *The Journal of Philosophy* 92, no. 7 (July): 345-381.

———. 1998. The dynamical hypothesis in cognitive science. *Behavioral and Brain Sciences* 21, no. 05: 615-628. doi:10.1017/S0140525X98001733.

Gödel, Kurt. 1992. *On Formally Undecidable Propositions of Principia Mathematica and Related Systems*. Trans. B. Meltzer. New York: Dover Publications, originally published in 1931.

Goldin, Dina, and Peter Wegner. 2008. The Interactive Nature of Computing: Refuting the Strong Church–Turing Thesis. *Minds and Machines* 18, no. 1 (March 1): 17-38. doi:10.1007/s11023-007-9083-1.

Goodman, Nelson. 1955. *Fact, Fiction, and Forecast*. Harvard University Press.

Haugeland, John. 1985. *Artificial Intelligence: The Very Idea*. MIT Press.

Haugeland, John, ed. 1997. *Mind Design II: Philosophy, Psychology, Artificial Intelligence*. Revised and enlarged edition. MIT Press.

Haugeland, John. 1998. *Having Thought: Essays in the Metaphysics of Mind*. Cambridge: Harvard University Press.

Hawking, Stephen. 2001. *The Universe in a Nutshell*. Bantam Books.

Hutchins, Edwin. 1996. *Cognition in the Wild*. MIT Press.

Kauffman, Stuart. 1993. *The Origins of Order: Self-Organization and Selection in Evolution*. Oxford University Press.

Kleene, S. C. 1936. General recursive functions of natural numbers. *Mathematische Annalen* 112, no. 1 (December 1): 727-742. doi:10.1007/BF01565439.

McDowell, John. 1996. *Mind and World*. First paperback edition with a new Introduction. Harvard University Press.

Milner, Robin. 1989. *Communication and Concurrency*. Prentice Hall.

Port, Robert, and Timothy van Gelder, eds. 1995. *Mind as Motion: Explorations in the Dynamics of Cognition*. MIT Press.

Putnam, Hilary. 1988. *Representation and Reality*. MIT Press.

Searle, John. 1980. Minds, brains, and programs. *Behavioral and Brain Sciences* 1: 417-424.

Searle, John R. 1992. *The Rediscovery of the Mind*. MIT Press.

Simon, Herbert A. 1962. The architecture of complexity. *Proceedings of the American Philosophical Society* 106, no. 6 (Dec 12): 467-482.

Smith, Brian Cantwell. 1996. *On the Origin of Objects*. MIT Press.

———. 2001. Situatedness/Embeddedness. In *MIT Encyclopedia of Cognitive Science*, ed. Robert A. Wilson and Frank C. Keil. MIT Press.

Stein, Lynn Andrea. 1996. Interactive programming: revolutionizing introductory computer science. *ACM Computing Surveys* 28, no. 4es (12): 103-es. doi:10.1145/242224.242358.

Turing, Alan M. 1936. On computable numbers, with an application to the Entscheidungsproblem. *Proceedings of the London Mathematical Society* s2-42, no. 1 (July): 230-265. doi:10.1112/plms/s2-42.1.230.

Wegner, Peter. 1997. Why interaction is more powerful than algorithms. *Communications of the* ACM 40, no. 5 (5): 80-91. doi:10.1145/253769.253801.

Wegner, Peter, and Dina Goldin. 2003. Computation beyond Turing machines. *Communications of the* ACM 46, no. 4 (4): 100. doi:10.1145/641205.641235.

Wolfram, Stephen. 2002. *A New Kind of Science*. Wolfram Media.