What Do We Talk About, When We Talk About Computing?[1]

Fodor, Formality and the Philosophy of Mind

One spring day, in the early 1970s, I stepped out of the MIT Artificial Intelligence Laboratory, where I was a graduate student, crossed the (literal and metaphorical) tracks on Main Street, and entered Building 20, home of the MIT philosophy department. I had an appointment with Jerry Fodor, a man I'd never met. "How do know computers work in the way you say?" I asked him, having read a draft of "Tom Swift and his Procedural Grandmother," and feeling that its picture of computation didn't comport with what had ended up in my blood and bones, after five years of developing programming languages and real-time operating systems. "Oh, I just made it up!", Fodor said—with what I would of course soon recognize to be characteristic gusto. "No, really," he went on; "we philosophers don't know much about these things. You're the computer scientist. Computers work in whatever way you understand them."

I don't recall what I said, but I remember what I thought. "If your understanding rests on my understanding, then you're in trouble!" But Jerry wasn't the slightest perturbed. "If it doesn't sound right, why don't you write a philosophically cogent account for us of how computers *do* work," he offered, ever helpful. Now famous people should be careful saying things like that. Sentenced me to 25 years of hard labour, he did, with just that one sentence. But that's okay. Twenty five years is up. I've come back to make a report.

This paper has three parts. First, I want to say a quick word, by way of review, of how computation is traditionally viewed within the philosophy of mind (how it allegedly works, and what problems it's taken to solve). Second, I want to lay out, as best I can, what computation is really like. Third, I'll speculate briefly on how that revised account should impact how we view the mind. Time, unfortunately, will restrict the third section to a few quick remarks, which is too bad, since it would otherwise have been the most interesting. But that's okay; fair's fair, in the long run. I'll leave it Jerry to explain the consequences of the new view for philosophy in his comments.

1. Introduction

Here's how it traditionally goes. To start with, there is a question of how any real naturalisable process can: (a) on the one hand be semantic or intentional—be rational, be coherent under interpretation, make semantic sense; and (b) be plainly and ordinarily physical, in two senses: first, be able to interact with the physical world—*cause* things and be causally affected by things in appropriate ways (so as to solve a generalized version of the age-old problem of mental causation); and second, be *made* of physical stuff (so as to satisfy

---

[1] With apologies to Raymond Carver. This paper is a rough précis of (part of) a forthcoming book on formal symbol manipulation, entitled *The Age of Significance: Volume II: Formal Symbol Manipulation*. Thanks to Irene Appelbaum for helpful comments on an early version.

a basically materialist metaphysics). Thinking, at least thinking rationally, evidently had the former property, of making semantic sense, but no one knew how it worked (even if we knew *that* it was physically possible). Written inscriptions, on the other hand, like the notes Holmes left for Watson, had the latter property, of being unproblematically physical, and even of making sense under interpretation; but they were inert. They didn't *do* anything.

Turing's insight was simple: plug the sentences in! Make them stand up and dance! That is: give physical inscriptions sufficient causal agency to make them interact and produce behavior. How to interact? In virtue of what we like to call their *form*—that is, various (supposedly unproblematic, albeit rather abstract) physical features, like shape and arrangement. If you set things up just so—get the physical / formal properties to correspond to the semantic properties in the right way, and define the rules of interaction appropriately (which is sometimes possible, though non-trivial; that too was one of Turing's achievements, taken from Frege and others in the logical and meta-mathematical tradition)—then it seems as if you can have just what you always wanted (if you were a basically physicalist philosopher): something like a naturalistically sound explanation of rational thought. Moreover, as Fodor is fond of reminding us, you also seem to get something about which no one has the remotest possible alternative explanation.

So that's computing, as traditionally conceived: metaphysically unproblematic arrangements of interpreted physical symbols, manipulated in virtue of their form so as to make semantic sense.

Needless to say, there are a few problems. Those semantic properties, the ones that the formal properties are supposed to be defined in accordance with, or at least defined to honour —it is not one hundred percent clear where they come from. More specifically, it is not clear whether they are supposed to be part of the computational story, in which case the theory of mind could be expressed by the slogan "mind = computing," or whether the "computational" part has only to do with the those (again: allegedly metaphysically unproblematic) formal / physical features and the behavior that results from them being manipulated, leaving the semantics for someone else to worry about. Most philosophers seem to lean towards the latter division of labour, with the result that a theory of mind is thought to need two parts:

1) A *computational* part, to explain how things are physically implemented—how they *work*, as it were; and
2) A theory of mental *content* (equivalently: *semantics* or *reference*), to explain how those thoughts have the interpretations that they do.

In "Methodological Solipsism," Jerry made a big deal of arguing that, for purposes of scientific psychology, we could only hope for the former. But then, with more of that trademarked gusto, he proceeded to ignore his own advice and spent the next twenty years of his life trying to formulate a theory of the latter.

2. Four Clarifications

Before pressing on details, I want to say a couple of preparatory things, by way of clearing the land. To died-in-the-wool philosophers of mind these remarks may seem a little distracting; but I believe that they are necessary in order to understand how the view just

introduced relates to anything recognisable by a practicing computer scientist. There is no *necessary* relation between Turing's "Good Idea" and the ensuing computer revolution, of course. Fodor would have given Turing tenure even if Microsoft and Intel had never happened. On the other hand, it would be bizarre if Microsoft's and Intel's wares worked according to a different principle than the one about which philosophy is so excited. If nothing else, it would apparently falsify Fodor's claim that we have no other ideas about how semantic things could work. But more seriously, I am simply assuming that it is worth philosophy's time to know how what I call "computing in the wild" does in fact work; that, after all, was the original twenty-five year project. So I ask for a few moments indulgence.

Four comments in particular.

2.a. Engineering

First, from the fact that we make computers—design them, debug them, build them, use them, repair them, sell them, modify them, and the like—it does not follow that we know how they work. Computer science is first and foremost an engineering practice, and a pretty pragmatic one at that. Its methods are not so much either empirical or theoretical as they are *synthetic* (a fact, I believe, of considerable significance for the philosophy of science[2]). [3] Dretske may be right that you can't understand something if you don't know how it works; but the converse doesn't follow: just because you *do* know how it works, that doesn't mean you understand it. I am going to take extant computational practice very very seriously. But that doesn't mean that I am going to defer slavishly to the *discourse* of computer science—to what computer science *says* about its practice.

2.b. Different construals

Second, the picture of computing that I just painted—what I will call the "formal symbol manipulation" idea about computing—is by no means universally accepted. More seriously, and perhaps more surprisingly, it is not even all that widely recognized. It is just one instance of half a dozen *construals* of computing: suggestions or stories about the essence or fundamental nature of computing. The formal symbol manipulation (FSM) construal is by far the best-known within artificial intelligence, cognitive science, and the philosophy of mind. But there are others: such as the idea that computing is *information processing*, that computers are *digital state machines*, that to be a computer is to *follow a rule* or *execute an instruction*, that to be a computer is to be *equivalent to a Turing machine*. Not only are these manifestly different in meaning; they are at least arguably in extension as well. The notion of a computer being a digital state machine makes no reference to any semantic, intentional, or even formal properties, for example; whereas nothing about the formal symbol manipulation view just articulated (neither on the surface, nor down deep, either) requires that it apply only to the digital case. Moreover, to make matters worse, each construal has versions: the in-

---

[2] Faced with a divergence between their ideas and reality, programmers are far more apt to respond by debugging reality than by modifying their ideas—without even ironic regard for the seeming travesty to the Empirical Method.

ref

formation-processing construal forks into at least three or four different ideas, depending on your conception of information (e.g., a syntactic one, descendent from Shannon and Weaver, e.g. by way of Kolmogorov and Chaitin; a semantic one, e.g. through Dretske and Barwise and Perry to Rosenschein and Halpern; and whatever notion of information undergirds the "information superhighway").

Most surprising of all, the FSM construal *is not the construal of computing that underwrites current theoretical computer science.* That is a strong statement, which I won't even try to defend here (except to say that figuring it out cost me five of those twenty five years). The point is just that the theory of effective computability, complete with the notion of a Turing machine, of Turing-equivalence, and the whole resulting body of work including halting and completeness results and characterisations of computational complexity—i.e., both absolute and relative computability results—is based, I am convinced, on a different set of intuitions from those in focus in the philosophy of mind. It is not particularly concerned with the sorts of semantic interpretation that infect the FSM account; misleadingly, its has its own distinct notion of "formality." Needless to say, no theory can lay claim to being a "full" or "final" theory of computing without figuring out how these two bodies of theory interact (I myself will point towards such an integration by the end of this paper). But that does not mean that the two construals say similar things about the same phenomena, or even that they give voice to comparable intuitions. As a result, theoretical computer scientists won't much recognize any of the considerations being wrestled with in this paper.

2.c. Program vs. process

Third, something that considerably complicates the investigative situation (especially these differences among competing construals) is that several of them use the same theoretical terminology to talk about different things. No case is more distracting than the notion of "semantics." I just said that the dominant conception of computing in theoretical computer science doesn't deal with the sorts of semantical issue that preoccupy the FSM idea and the philosophy of mind. It does, however, talk about "semantics"—but (unfortunately) to refer to a different phenomenon.

A simple way to explain this is shown in Figure 1. Computer science is interested in *behaviour*—in things working, in getting things done. Programs, per se, are not behavior. They are static, usually textual entities that one edits on a screen, prints out on paper, etc. A great deal of computer science's focus is squarely on programs—but not, at least not just, as passive, even static, lexical structures. Rather, programs are written to be *executed*, to *run*, to give rise to *behaviour*, to engender *processing*. It is always that processing, that behavior, that is of paramount concern.

Thinking is a process, too. And so you might well think (and should think, according to me) that the appropriate analogy between computing and psychology would be to draw parallels between thinking, in the human case, and the processes that result from executing programs, in the computer case. I.e., the analogy should be between mind and *process*, not between mind and *program*.[4] Moreover, computer science uses the term 'semantics' to refer to the *program–process relation* indicated as 'α' in the diagram. if asked about the semantics

[4] This was one of the troubles I had with Fodor's 197… paper.

of some construct C in a programming language L, the proper computer science answer would give an account of *the behaviour that executing that construct will lead to*. The semantical question in psychology, however—or at least the semantical question that exercises philosophers—is not, in the first instance, a question about a program–process relation α, but about the relation between a *process* (thinking) *and its subject matter* or outside world—i.e., the relation labelled 'β' in the diagram. The only way that I have ever been able to explain, to working computer scientists, what we in philosophy of mind mean by a "theory of semantics" is something that computer scientists, if they called it anything, would call *"the semantics of the semantics of a program.*" Here, I will simply call it "process semantics," to be distinguished from "program semantics."

This semantical ambiguity has been obscured by the fact that computer science currently has no canonical way of specifying behavior or processes. So computational processes, which we philosophers would call intentional, are typically described or modelled, in theoretical computer science, *in terms of (what we philosophers would call)* their content. Thus consider a process of adding numbers—or rather, as philosophers would strictly say, if pressed, of *adding numerals*—or even, to be maximally pedantic, of *adding* (genuine) *numbers by crunching numerals*. Computer scientists would simpy call that *adding*, and pretty much deny that numerals are involved, because the term 'numeral' (to them) will signify *program* elements, and we are talking about *real behaviour*, not lexical syntax.

Enough complexities. I am simply making two points: (i) to the extent that computer science is interested in form, syntax, etc., it is *programs* that they view as syntactic or formal, not the *processes* to which they give rise upon execution; and (ii) the program-process relation (α) is the relation called "semantic." Philosophy of mind, in contrast: (i) by and large doesn't distinguish programs from processes; and (ii) thinks of the process-world relation (β) as the paradigmatic semantical relation. Because of these two differences, serious discourse between the two camps has been virtually non-existent.

2.d. Compositionality (or the lack thereof)

The fourth introductory remark follows from the third, and touches on recent debates, particularly having to with connectionism, about the systematicity and productivity of computational models.

Some writers have assumed that all programs written in the "von Neumann" style—that is, in terms of explicit, lexical programs—must be instances of a "symbolic" or "logical" or "conceptualist" variety. Other models of computing—notably connectionist networks, but presumably networks of other varieties as well, to say nothing of cellular automata, Petri nets, and who knows what other ilks—are thought to be *non-symbolic*, and thus not to depend (or anyway not to depend so blatantly) on conceptualised representation. Some take that to be an advantage; some, a disadvantage. Fodor and Pylyshyn, among others, have

---

I have a lot of personal experience with these problems. In my 1982 doctoral dissertation, I defined what I thought of as a particularly semantically clean programming language, based on our (philosophical) conception of semantics (i.e., in which the semantic relation of interest was the process-world relation labelled 'β' in the figure). The language itself was not ignored; conferences based on that work are still being held on a roughly annual basis. But the semantical stance I worked so hard to adopt never made a shred of sense, from a "computational" (i.e., theoretical computer science) point of view.

strongly defended the "language of thought" (conceptualist) model, arguing that no other architecture can possibly explain what any adequate psychology must explain: the systematicity and productivity of thought.

For the moment, I want to point out just one striking feature of this debate: it seems to compare GOFAI-like knowledge representation systems, on the one hand, of which perhaps a thousand have been built, with various more modern connectionist alternatives, extant numbers of which perhaps also number in the small numbers of thousands. They entirely ignore the *vastly* more common code (who knows? perhaps one hundred billion—10^{11}—lines of C++?) that underwrites innumerable commercial programs, from word processors to washing machine controllers to e-mail systems to internet routers to drive-by-wire automobile control systems. Your typical 1990s washing machine, to pick an example at random, has about 500,000 lines of code in it, using as many as 100,000 different identifiers. It seems unlikely (perhaps fortunate as well) that the conceptual prowess of a washing machine is to be measured in the same number of concepts as are to be found in the *American Heritage Dictionary*.

The explanation is not hard to find. What is symbolic—compositional, systematic, and productive—in ordinary programming practice is the *language in which individual programs are written*, *not the programs that result*. All those 10^{11} lines of C++ (suppose it amounts to a hundred million different programs) are written in the *same* language. Sure enough, you can mix and match and combine arbitrary fragments of C++ in pretty much any way you please —that is why it is a general programming language. But once the program is written, that "mixing and matching" is done. Finito! It is extremely rare, to the point of non-existence, for the resulting programs themselves (or processes to which they give rise) to have any compositional prowess with respect to their resulting data structures. The washing machine will have one variable that counts the number of seconds that the oscillator has been running, and another to monitor the temperature on the motor coil. It can recognise a surge in voltage, and a delay in washing—but not a surge in washing, or a delay in voltage. I.e., in spite of being written in a "von-Neumann" language, it no more satisfies an Evansian generality constraint than does a connectionist network. That is: no more compositional prowess should be attached to a program just because it was written in a compositional programming language than symbolic powers should at attributed to a car engine because it was designed on a CAD system. You don't need a network not to be conceptualist; all you need is something just about every actual working program in the world already has.

In case you were wondering.

3. Formal Symbol Manipulation

Return, then, to psychology, formality, and the philosophy of mind. As I see it, there are three essential conceptual ingredients in the formal symbol manipulation construal (FSM) of computing.

Many "names" will be repeated, but used in highly-regulated different contexts so as not to cause conflicts.

3.a. Semantics

The first, betrayed by the word 'symbol,' is that FSM takes computation to be an ineradicably intentional phenomenon. It is crucial to recognize this fact up front, because various readings of the second essential ingredient in the FSM construal, the notion of *formality*, will attempt to hold that semantical aspect of computation at bay. Thus we will see readings of formality that claim that the semantical properties are derivative, that they play no role in engendering behavior, that they can be changed without affecting the syntax or mechanism, that accounts of how computers work can be given without reference to them. But to separate something out, or be independent of it, or even to ban it, is not to deny it its existence. On the contrary: in order for semantical properties to be held at bay in these various ways, *it is a logical prerequisite that they exist*. And so the formal symbol manipulation[7] construal of computing is unavailable for intentional or semantical eliminativists. The situation is thus more like the prohibitionists' rejection of alcohol than it is like the physicists' rejection of the luminiferous æther. Just as alcohol must exist, in order for a prohibitionist stance to be substantial, so too semantics must exist—must be an essential property of symbols—in order for the formalists' setting of it aside to count as a substantive position.

Take it as an issue of truth-in-advertising. If some form of semantics were not an essential property of formal symbol systems—derivative or original, intrinsic or attributed, or at the very least *attributable*—there would be no reason for the term 'symbol' to appear in the label. In fact I will dub any attempt to construe computing as not involving semantics at all with the simpler and more honest label: *stuff manipulation*.

It follows[8] that it is a mistake to equate the terms 'computational' and 'syntactic.' Some writers seems to think that cognitivism can be described as a position according to which mental representations exemplify two kinds of properties: *computational* properties and *semantic* properties. But *if you are committed to an FSM construal of computing,* that is an elementary mistake. On a cognitivist line, mental representations are *computational*, which means that they exhibit (on the traditional view) two kinds of properties, syntactic and semantic.[9]

So symbols are ineleminably intentional. Can anything else be said about them. In particular, can we say anything about the *kind* of sign or representation or signifying structure that a symbol must be? In brief: no—not if the FSM construal is to stand a chance of serving as a general characterisation of computing. This was implicit in those ealier comments about a hundred billion lines of C++. Again, this does not seem to be adequately recognized. Some writers seem to think that, among potential kinds of representation, including perhaps implicit and distributed, the *symbolic* is a special kind—one, perhaps, that treats representation as necessarily conceptualist or explicit (in a way, perhaps, that traditionalists have taken to be cognitively essential, but that these writers view as perhaps artifactually linguistic, not

---

[7] Theorists of such predilection should instead be interested in the other construal: of a digital state machine, for example, or the Turing-theoretic notion of effective computability

[8] refs?

[9] Or at least *interpretable*. Should anything be said about the potentially dispositional nature of the semantic interpretability?

particularly cogent for the purposes of fundamental psychology, etc.). But even if symbolic computation of that sort is possible, it is at most a rare species, unlikely (to repeat a point made above) the vast majority of functioning software. If we are to even begin to have a chance to capture what I call "computation in the wild," then, the term 'symbol' must be read very widely—to include all sorts of implicit and explicit data structures, codes, information, knowledge, pictures, data bases, records, stories, language, icons, descriptions, blueprints, and so on and so forth, without much apparent limit.

What we've said so far, in sum, is that (i) 'symbol' must be read semantically, but (ii) otherwise with little or no constraint.

3.b. Formality

Turn, then, to the second essential ingredient in the FSM construal: the term 'formal.' Most of the weight of the construal is borne by this qualifying predicate.

Why so? In part, I believe, the motivations for believing in a notion of formality—for thinking that psychology should take computation seriously—have to do with a vague but strong intuition that formality will confer onto the project of understanding the mind some major naturalising benefit. That is, it is widely assumed that formality conveys onto the symbolic or semantic aspect of computing a cleanliness or tractability it would otherwise lack—a cleanliness presumably denied to other "non-formal" intentional processes (including us, perhaps, if cognitivism is false). In spite of being assumed in the background to be ineliminably intentional, in other words, computers are somehow thought to be *rescued, in virtue of their formality, from the <u>really</u> difficult questions of semantic existence that plague other intentional phenomena*—such as about how reference arises, or what establishes the conditions for semantic success, or how truth relates to goodness and virtue. Formality, that is, is taken to ensure computers a safe, middle-ground existence—above the drudgery of mere brute mechanism, but safely below the full-blown (and perhaps uniquely human) citadel of consciousness and semantic originality.

What does 'formal' mean? It turns out that there are two readings, one positive and one negative.

On the positive side, the behaviour or operations or manipulations of a symbol system are most commonly described as formal just in case their manipulation can be shown to depend solely on the *syntactic* or (as it is said) "formal" aspects of the ingredient symbols. It is a very familiar intuition. We all know the catechism: if a computer moves from '2+3' to '5', or from '$\forall x$ [MAN(X) $\supset$ MORTAL(X)]' and 'MAN(SOCRATES)' to 'MORTAL(SOCRATES)', then according to the positive reading that transition is formal just in case it can be achieved solely in virtue of the syntactic properties of '2', '3', 'MAN', '$\supset$', etc. In fact on the positive view *syntax* and *form* are typically identified: for obvious etymological reasons, the properties on which the behavioural mechanism can depend are often simply called its formal properties, as in: "logical inference regimens can only depend on *formal* properties of the constituent expressions."

But familiarity is not clarity. A hint that there may be confusion under the surface is betrayed by the number of different words that appear, in the literature, for essentially this one positive aspect: *syntax* (whatever syntax turns out to be), *grammar*, *shape*, or, as I have said, *form*, or a subset of the *physical* or *causal* or *mechanical* or *effective* or *potent*—or even,

sometimes, *local* or *intrinsic*. What a tangled web! A host of questions need to be raised: about what each of these eleven terms means; about whether they are individually coherent; about whether they refer to exclusive, or overlapping, or identical, properties; about their naturalistic or other explanatory status; about the metaphysical baggage they carry (think of what is implied by the use of 'intrinsic,' e.g.); etc. The issue of whether syntactic properties are physical, for example, or are only physically realised, or are not physical after all but instead perhaps wholly abstract, contains within it the nub of the very mind-body problem that the formal symbol manipulation construal is alleged to help us resolve.

For discussion, and for reasons that will emerge a little later on, I will refer to the positive properties in virtue of which a formal symbol systems works as its *effective* properties—i.e., will (for the moment) take the term 'effective' as a gloss or summary for the list of eleven just given, until such time as we have a better theoretical understanding of just what they come to.

The negative reading of formality is understood as a restriction—as a condition imposed on generalized symbol manipulation which holds that the operations of the system must proceed *independent of the* (symbols') *semantics*. I.e., the innards of a formal symbol manipulating machine are assumed, on this negative reading, to operate, doing whatever they do, in utter disregard for such messy semantical predicates as truth, meaning, and reference. Intuitively, that is, the eventuating behaviour or yielded-up result, in the examples just given, is required, on the negative reading, not to interact with (whatever that would be to do), or otherwise to depend on: Platonic numbers, the truth of universal claims, or Socrates himself, manhood or mortality, or any other "real-world" entities that figure in the claim's meaning of content or interpretation—or, for that matter, on any of the facts that establish or assign these entities as the symbol's interpretation or semantic values. All these external facts—even whether Socrates existed—are on the negative reading of formality claimed to be *mechanically invisible*, as behaviourally irrelevant to the system as the truth of the continuum hypothesis is irrelevant to schoolyard children playing hopscotch.

All computers do, in sum, according to this negative version of the story, is to move the symbols around—blindly, as it were, in blissful ignorance of the stuff in the world that the symbols are about.

It is this negative reading that Fodor has dubbed the "formality condition," and which for discussion I will refer to as the "antisemantical" or "negative" reading.

It is evident that a number of background conditions must be assumed to hold, in order for this double (positive and negative) characterisation of formality to work—beyond the non-trivial fact, already mentioned, that some form of semantic interpretation or interpretability for the symbols is necessary, in order for the negative claim to get any conceptual purchase.[10] For example, it must also be assumed that the potent properties fall outside the scope of the semantic ones, or else the two readings, positive and negative, would

---

[10] Consider chopping up eggplants for mousakka. It is a stretch, but perhaps marginally coherent, to imagine that you could do so formally on apositive reading of formality: that is, slice them in virtue of their physical shape. But you can't chop eggplants formally on a negative reading for formality, for the same reason that prohibitionists couldn't complain about alcohol if there weren't any alcohol: chopping eggplants independent of their semantics would require semantics to be independent of; and eggplants don't have semantics, at least not qua moussakka ingredient.

pick out different systems—i.e., would be extensionally distinct. Since none of the eleven characterisations used to characterise effectiveness (*syntax*, *grammar*, *shape*, *form*, *cause*, *physical*, *mechanical*, *effective*, *potent*, *local*,[10] or *intrinsic*) *mean* non-semantic, however, this is a substantive thesis, not a tautological one. As such, it is subject to critical assessment.

3.c. Manipulation

The third essential property of the FSM construal is implicit in the third word in its name: 'manipulation.' Namely: the claim that is made—the implicitly esential criterion on a computer's being formal—has to do with *how the computer works*. This mechanical emphasis underwrites the entire "mechanistic" tradition from which computation sprung; and it is such an evident or perhaps underlying aspect of the thesis that it isn't often raised into explicit theoretical view. But it matters, as we will see. In particular, the implied thesis is something like the following:

> To be a computer—i.e., the defining or essential aspect of being computational—is to be composed of symbols, on a liberal conception, such that the way that the computer works (mechanically, effectively, causally) is via the formal manipulation of those symbols.

Formal, in the last phrase, is then taken in both senses: positively, to mean something like *grammatical* or *effective*, and negatively, also, to mean something like "independent of semantics".

The reason that it is important to highlight the emphasis computational "working" is that, unless one is careful, it is easy, in attempting to understand the positive reading of formality, mentioned above, to generalize it to the point where the overall thesis is rendered vacuous. In particular, it is clear that the notion of "works" isn't so inclusive as to mean "has as its complete nature"[11] —for that would be contradictory. A symbol cannot, as its complete nature, be independent of the semantics that constitute it. More particularly: it is only if the term "works" in the characterisation of FSM just given means something *less* than "has as its complete nature" that the *negative* reading of formal symbol manipulation makes any sense. On the other hand, the *positive* reading of the claim would be vacuous if it merely came down to a statement that the effective aspect of a symbol system is only constituted of its effective aspects, or that the causal behaviour of a computer can only depend on the causal properties of its (causally-individuated) parts. Substantial claims require a kind of "double access" to the subject matter: we must first take hold of the symbols and operations—with our left hand, as it were—and then place conditions on them by bringing to bear a different set of properties—properties grasped, as it were, in our right hand.

---

[10] Strictly speaking, only seven—*grammatical*, *syntactic*, *causal*, *physical*, *mechanical*, *intrinsic*, and *effective* are higher-order properties, and thus have properties in their extension; the same is not immediately true of the other two—*shape* and *form*. On the other hand, higher-order versions of these can easily be constructed, so that *being triangular*, for example, is taken to be a "shape" predicate.

[11] In the sense, for example, in which one might ask how it "works" that the sum of the digits of the decimal representation of any number divisible by 3 will also be divisible by 3.

3.d. Summary

In sum (since we need to move on), formal symbol manipulation can be summarised in terms of the following tri-partite typology. Distinguish three types of property that a symbol can or will exemplify:

p1. *Syntactic*: those involved in what might be called a system's grammatical regularities;

p2. *Effective*: those in terms of which the mechanical operation (i.e., the "workings" of the system is defined—i.e., that play a causal role in engendering behaviour; and

p3. *Semantic*: such as truth, reference, meaning, etc.

The positive reading of the formal symbol manipulation claim can then be taken as claiming that the syntactic properties are the effective ones:

positive: $p_2 = p_1$

The negative reading, in contrast, can be understood as a claim that the effective and semantic do not overlap:

negative: $p_2 \cap p_3 = \varnothing$.

And the assumption that the two readings, positive and negative, come to the same thing (extensionally), in any or all instances, depends in part on what I will claim is perhaps *the most universal and unquestioned claim in the intellectual vicinity*: that a formal symbol systems' syntactic and semantic properties are disjoint:

disjointness: $p_1 \cap p_3 = \varnothing$

It is too early, given the state of theoretical art, to say whether any of these assumptions are necessary, contingent, or even true. That would require theories of syntax, semantics, and effectiveness—a triple demand, of which we presently have none. But the basic conceptual structure is implicitly recognised. Thus here is Fodor, in "Methodological Solipsism":

> What makes syntactic operations a species of formal operations is that being syntactic is a way of *not* being semantic. Formal operations are the ones that are specified without reference to such semantic properties of representations as, for example, truth, reference, and meaning. Since we don't know how to complete this list (since, that is, we don't know what semantic properties there are), I see no responsible way of saying what, in general, formality amounts to. The notion of formality will thus have to remain intuitive and metaphoric, at least for present purposes: formal operations apply in terms of the, *as it were, shapes* of the objects in their domains. [Fodor, *Methodological Solipsism*, emphasis added]

Fodor thus first identifies formality negatively, but then takes the positive (syntactic) reading to be a *species* of nonsemanticity, implicitly relying on the near-universal assumption expressed above (p3), that syntactic and semantic properties do not overlap. By the end of the

passage, however, Fodor has shifted to equating formality with effectiveness—recognizing it, in his hedged reference to "as it were, shapes," to be an as-yet unreconstructed term.

For our purposes, it is important to analyse the two readings independently. At first blush, one might think that the positive (syntactic) reading would be the more constructive, since it apparently identifies an actual set of mechanically potent properties on which system's behaviour can operationally or effectively depend—namely, those that play a constitutive role in the system's grammatical regularities. Ultimately, however, I will argue that the intuition underlying it will to receive its most natural reconstruction not under an analysis of formal symbol manipulation at all, but within the context of the recursion-theoretic notions of complexity and computability, mentioned above as more central to theoretical computer science—that is, under a construal that *does* take computing, after all, to be neither more nor less than "stuff manipulation."

For now, I want to keep the negative reading in focus: the one having to do with the distinctness, separateness, or independence of the semantic. This is the reading, I will claim, that is traditionally viewed as lying at the heart of the formal symbol manipulation thesis. For as already mentioned, it is exactly the alleged freedom from semantical considerations, explicitly mandated by the negative reading, and implicitly suggested by the background assumption that syntactic and semantic properties do not overlap, that is thought to convey on formal symbol systems the conceptual cleanliness they are thought to enjoy—no threat of irreducible social ties, no complex questions of interpretation, no room for divine intervention—that so recommends them for a philosopher interested in naturalizing the mind. The negative reading of formality, that is to say, seems to offer naturalists their dream on a silver platter: a clear and direct manifestation of how intentional processes (as has been said, such systems *are* symbolic, by hypothesis) can unproblematically be realised in a physical substrate (no mind-body problem, either).

If only it were true.

4. Conceptual Critique

There are three fundamental problems with the FSM thesis. Conceptually, it is confused. Explanatorily, it wouldn't be naturalistic, even if it were fixed. And empirically, it is false. Otherwise, it's in fine shape.

I want to consider the conceptual problems first. Unfortunately, they're a bit abstract, but they are of foundational importance. Moreover, if we can make our way through them, we can move into the (relatively easier) sailing of showing how the claim fails to be true of concrete practice. Look forward to that as something of a dessert.

The conceptual problems stem from the fact that the negative reading of formality is systematically ambiguous [see #n on the handout]. That is: there are two versions of what "independent of semantics" means. The first operates at the level of theory or concept, and is reminiscent of a reductionist desire for a "semantics-free" account. It takes the fsm thesis as a claim that computation can be *described* or *analysed* in a semantics-free way. If that were true, or so the argument goes, that would go some distance towards naturalizing intentionality (as Haugeland, primary adherent of this interpretation, says, it would help "make the world safe for semantics"). The second motivating intuition, different in character, operates

one level down, at the level of the phenomenon itself. It is essentially an empirical claim on the ways that computers actually work. Again, the motivating idea is familiar: a claim that intentional phenomena, such as reasoning, hoping, or dreaming, carry on in relative independence of their subject matters or referents. Reference and truth, it is recognized, are just not the sorts of properties that can play a causal role in engendering behavior—for perhaps quite difficult reasons, but essentially because they involve some sort of relational coordination with things that are *too far away* to make a difference.

The problem is not just that these two readings are different. They are in outright conflict. To the extent that one is true, the other is false. Moreover, they would both need to be true, simultaneously, for the naturalist's dream to be realized. So the issue is far from minor.

conceptual: Computation can be *characterized without reference to* semantical properties.

empirical: Computation *works independent of* semantical properties.

4.a. Conceptual reading

What I will call the "conceptual" reading [listed on the handout] claims that computation can be *characterized without reference to* any semantical properties. This sense (among others) figures in Fodor's characterisation, above. "Formal properties," he says, are ones that can be "specified without reference to such semantic properties of representations as, for example, truth, reference, and meaning." The idea is again familiar from the logical catechism: by and large one spells out the syntax, the terms, the identifiers, the formation rules, and the whole inference regimen, before (and thus without relying on) any specification of the semantics.

Why the semantic ascent? It is hard to be absolutely definite, but surely the answer is essentially this: *because it seems to comport with a naturalization projects*. Why so? Well, think about what it would be to successfully naturalise some problematic phenomenon α, such as intentionality, or consciousness. What you want is a characterisation fitting something like the following schema:

α = such and such

where the term on the left—the explanandum, α—is (a term designating) the potentially problematic phenomenon in question, and the description on the right is an explanation in naturalistically palatable terms of that same phenomenon. The description on the right, that is (to be absolutely pedantic), has two crucial properties:

1) It describes the phenomenon in question; (that's what makes it a theory *of* α); and
2) It is expressed in *naturalisatically palatable language*.

So if we say that heat is mean molecular kinetic energy, then the (otherwise undefined or unanalysed) term "heat" occurs on the left, qua explanans; and the description "mean molecular kinetic energy" occurs on the right, qua explanandum. This theory is is legitimate naturalistic theory of heat because the explanandum, "mean molecular kinetic energy," meets both criteria we wrote down: (i) it is a description of heat, both necessary and sufficient, we may presume; and (ii) it is expressed in naturalisatically palatable language, because notions

like "kinetic energy" and "molecular" receive their interpretations from the lower-level grounding theories.

Suppose we had a naturalistic—indeed, reductive—account of computing, where computing was, as we are assuming it is, genuinely intentional or semantic. (Note: I realize we don't have such a theory; and I am not claiming anyone *thinks* we have such a theory; I just want to make a few points about what it would be like if we *did* such a theory.) Then we can presume that it would have that form:

computing = … such and such (γ)…

where "such and such", or γ, would be a description, in the terms of the lower-level reducing theory, of what computing, *including* (ex hypothesi) *its semantical aspect*, was like. If this was a complete theory, γ would designate the entire phenomenon of computing, *including its semantical aspect.* But it would do so in a way that was *free of any undischarged semantical predicates* (that's why it would be a *reducing* theory).

That is: we would have an account of computing that would *specify it*—what it was to be computational—*without reference to any semantic properties* such as truth, meaning, etc. But note how crucial is the word "reference." γ would *refer* to semantical properties—or at least designate phenomena necessarily co-extensive with semantical phenomena. The only [13] "freedom from semantics" it would exhibit would be in terms of *how it designated them*.

One final preparatory note. Although I pointed out that Fodor's characterisation of formality shades into this kind of conceptual vocabulary, the most prominent defender of a conceptual, "specified without reference to," reading of "independent of semantics" is John Haugeland. In *AI: The Very Idea* he painstakingly lays out a definition of "formal" that makes no reference whatsoever to any intentional notions at all: not to syntax, nor to grammar, nor to formulae, or anything in the vicinity. "Formal," Haugeland claims, means "finitely checkable, perfectly definite, … ." [14] That is: he provides is a paradigmatic naturalistic reduction of (we can presume) the positive reading of formality: what the features are in [15] virtue of which computers do, in mechanical fact, work.

4.b. Empirical reading

As I said, the other version of "independent of semantics" has a much more direct scientific or ontological character: it makes a substantive empirical claim about how computers work. That is [as indicated in the handout], it says that computers *work independently of the semantic properties that their ingredient symbols are assumed to have.*

The same point could be made—perhaps more legitimately—in terms of properties and types. I.e.: that the explanandum (the reducing characterisation) would refer to non-semantic properties of events (non-semantical because it is a reducing account), but properties of events such that, if any event manifests those properties, by force of the nomological law being expressed such an event would *also* manifest the requisite semantic properties.

Get this right.

That it is paradigmatically naturalistic in form does not mean that it necessarily succeeds, of course; it also needs to be *true*. Which I do not believe it is. For criticism see …

As will be evident, it was primarily this empirical version that I had in mind when explaining the negative ("independent of semantics") reading of formality in the previous section. This is also the reading that underwrites such statements as Searle's insistence that one can "change the semantics of a computer without changing the syntax." What it does is to take the independence of syntax (or effectiveness) and semantics to be a brute fact in the world, where situations of one type do not affect situations of another—rather in the way that the perihelion of Mercury does not affect the spelling of the word 'hors d'oeuvre,' or that riots in Los Angeles do not affect the sum of 2 plus 2. A condition on the subject matter, empirical independence is of a sort that could be empirically verified—by adjusting one entity (the semantics), for example, and seeing whether the other entity (the syntax) changed along with it. At heart it rests on a sense of uncorrelated variation, somewhat like that between a rectangle's length and width.

Thus suppose we say that the damage done by a projectile was a function of its kinetic energy, but independent of its colour. The idea, presumably, is that there are two *classes* of properties that a projectile can have: one of energy, another of colour. That is, a given projectile X (at a certain moment in time) may have one of a wide range of possible energies, and a wide range of colours—say, 142 joules and bright fuscia. The damage is done by (its having) the energy, and not (its having) the colour, because if it had been another colour—olive drab, say—but of the same energy, then the damage would have been the same; whereas if it had different energy—0.03 joules, say—but still been bright fuscia, the damage would have been much less. Modulo Humian debates about relations between cause and correlation, we can surely say that considerations of this sort of counterfactual correlation underwrite (or comport with, or whatever) our sense that the damage was *caused* by the energy, not by the colour.[16]

Thus the empirical version (of the "independent of semantics" reading) of formality rests on something like the following triple of assumptions, parallel in structure to what we saw above:

1) Computational symbols have syntactic or grammatical or shape properties (with any *given* symbol, at any location in space-time, having a *particular* syntactic shape, chosen from a range of possibilities);
2) Computational symbols have semantic or interpretive properties (with, again, any given symbol, at any location in space-time, having a particular semantic interpretation, chosen from a range of possibilities); and
3) The symbol is manipulated (i.e., its effective behavior results from processes that "work"):
   a. In virtue of the properties identified in (1);
   b. Independently of the properties identified in (2).

Readers of this paper should be relieved that I will not take up yet further distionctions, leading to sub-sub-readings, where the "independence" between these two kinds of property is in turn broken down into extensional (weak) and intensional (strong) varieties. For the full treatment, see the larger work cited in fn. 1.

It is (3.b), of course, that is the empirical version of the "independent of semantics" claim; (3.a) is simply the positive reading of formality, which we saw earlier (it is included here only for completeness).

4.c. Disconnection

Why should anyone believe this? Because of what I take to be the most profound semantical fact of all about intentional systems: that what we can think of as the "referential" or "semantic reach" of any interpreted or intentional phenomenon—the famous directedness that has been under central investigative scrutiny at least since (Chisholm's misinterpretation of [4]) Brentano—can and often does reach out across time, space, and possibility in a way that dazzlingly outstrips any mere causal relation. We can talk about the Pharoahs of Egypt, without invoking backwards causation; can refer to the first female President of the United States, without having to wait (God knows; another 50 years?) for our reference to succeed; can write novels, do a little speculative planning, talk about what would have been the case, if Cleopatra and Mark Anthony had won.

This **disconnection** of sign and signified is not only ubiquitous; it is so familiar that it is hard to imagine it's being false. It is what enables hypotheticals, allows us to plan and to dream, grants us memories of events long gone. Without it, fantasy lives would be banned, imagination cut off at the knees. The only way to think about tectonic drift would be to drag the plates along with you. All and all, it would be a dismal life—except that it would not be a life at all. Sans disconnection, existence would be restricted to nothing more than the suffocating press of $1/r^2$ forces that impinge on you within your proximal causal envelope.

Fodor has characterised the ability of symbols to continue to denote what they denote, in the absence of what they denote, their *robustness*. I think this term skews the priorities. We in the philosophy of mind sometimes take the paradigmatic case of semantic interpretation to be instantiated in *perception*: where you have a clear and distinct thought about a clear and distinct object that, without distraction, stands in immediate full view. That is: We in the philosophy of mind tend to consider cases where *signifying* X is *caused* by X—i.e., where X is causally present. We all know, though, that one can signify X in the absence of X. My crusade is to invert their status: to claim that the *disconnected* case is the *primary* case.

But leave crusades aside. The present point is just this: that at an informal or intuitive or pre-theoretic level, I believe it is the disconnection of (computational) symbols that underwrites our allegience to the empirical reading of "independent of semantics." True semantical properties, to put the same point another way, seem simply to be *too far away* to do any useful work in getting a brute computer to run. Suppose a theorem prover has to make a decision about what to do with an internal coding of the proposition that Alpha Centauri has 3 components, or is the brightest objectr in the Milky Way, or is 4.4 light-years away from earth. Among other properties, these claims have a *truth* property: of being right or wrong. The problem is, that truth property is determined a long way away from here. The computer has 5 nanoseconds in which to make its decision, before it has to move on. So it *can't* use truth

---

Ref Franchi.

If you're a computer, truth itself is just not available as the kind of property of your symbols you can rely on, as a way of doing business—especially not in the small. It wouldn't be good for your prospects of being picked up by Intel and marketed.

It is not even that semantical properties *could* be causally efficacious, if one had all the time in the world. At least often, they're not causally efficacious at all. Suppose you worked at the patent office, and someone submitted plans for a weird and wondrous gizmo, designed for people to wear, that claimed to be based on a new as-yet-undiscovered sensory principle allowing it to detect (and emit a warning beep) whenever the wearer was the subject was referred to—was the object of an intentional act. Not even the NSA can build meters like *that*. Acts of denotation don't bathe their referents with any form of discriminable energy—not even with a teeny wee bit; not with any *at all*.

So semantical properties—at least sometimes, and maybe even always—aren't causally efficacious. They're too remote, or too relational, or too abstract; something like that. Maybe they're causally efficacious, but simply not around for present use. Maybe semantical properties are causally impotent. But one way or the other, they aren't the properties in virtue of which computers work.

I will have more to say about disconnection later. For now, I only want to establish two things. The first is simply a suggestion that the empirical reading of "independent of semantics"—the idea that computers *work* independent of any semantical properties of their ingredient symbols—is intuitively based on, believe, and derives such truth as it has from, intuitions about disconnection.

Secondly, there is a connection between disconnection and independence. The reason that I motivated the "non-correlational" view of the (potential) causal impotence of sematnical properties is because … *Tai!*

4.d. Analysis

Note: I think the rest of this whole section (i.e., §s 4.d and 4.e) should go. Partly for theoretical reasons: I am taking on issues here that are not motivated by what preceeds. But mostly because it is vastly too "technical" and difficult. The question is what morals, if any (perhaps from the last couple of paragraphs?) should be extracted, before the rest is thrown away.

Now here's the problem. If, as we're assuming, computation is semantical, then a theory of computation has to include a theory of semantics. What our naturalistic yearnings suggest is that the *effective* part of computation: the workings, the "what it does, based on what is causally available to it" may in the end succumb to naturalistic explication. The problem is that the *success* of that naturalizing project depends on the *falsehood* of disconnection.

Think of it this way. The discussion can be summarised in terms of two sets of criteria that a theory of computation should meet. The first set places three substantive conditions on the coverage of the subject matter:

range: It should apply to all possible types (and instances) of the subject matter, not just to one specific architecture or kind;

aspect: It should explain all constitutive or essential properties or attributes, not merely a restricted subset or projection; and

level: It should treat these properties or aspects at the level at which the salient generalisations or regularities hold, not (or at least not only) at an underlying level of implementation.

Thus a generalised account of the syntax of symbol manipulation, applicable to all possible computational architectures, that nonetheless ignored semantics, would, because of the ineliminability of semantics, satisfy range, but fail to satisfy aspect. By the same token, even if we were to have laid out before us (the unlikely prospect of) a full analysis of the human condition, by hypothesis satisfying aspect, that account might still, as a full theory of intentionality, fail to satisfy range, because of applying only to people. Finally, an account of the underlying implementation of the syntactic aspect of a specific computational architecture would fail all three criteria.[18]

To these three subject-level criteria, we can add a fourth that those with reductive leanings will want satisfied:

m·naturalism: It should explain how it is that the (full) phenomenon in question is "of a piece" with the natural world as explained by the rest web of science.

Then we can identify a stronger one: the way of satisfying m·naturalism that would be preferred by those with with reductive leanings:

m·reduction: It should be formulated ("on the right hand side of the equations," as it were) in language free of undischarged semantical predicates.

How, then, do our readings fair?

As normally formulated, the empirical reading starts off badly: it fails both versions of the metatheoretic criterion. In fact it fails them doubly. For consider its logical form: it applies the restriction "works independent of semantics" to the larger encompassing "symbol manipulators." Since neither 'symbol' nor 'semantics' is explained, both filter and class fail m·naturalism. But perhaps that's a cheap shot. It is downright obvious, after all, that "operates on symbols independently of their semantics" is about as far from a naturalistically palatable explanation as you could imagine. Still, it is worth bearing in mind that, with respect our naturalistic yearnings, it makes not a whit of difference whether computers *mechanically rely on* their assumed-to-be-genuine semantic relations. Banishing semantics to

---

So, it should be said in passing, would a complete theory of the anatomy and physiology of the human brain. In spite of the fact that this prospect has excited many people in certain quarters of philosophy, and notwithstanding that such a theory might not only be extraordinarily illuminating in its own right, but even shed light on many profound questions of psychology, it is still important to understand that, as a theory of intentionality, a complete account of brain function would still fail to meet *a single one of the three substantive criteria*.

One way to meet the reductionist criterion would of course be to take computation to be a *semantics-free phenomenon*: deny that it involves symbol manipulation at all, and thus dismiss the idea that having an interpretation or being interpretable is an essential computational aspect. This would clearly be the favoured option of a semantical or intentional eliminativist. Some other distinguishing characteristic would have to be offered in place, of course, but other possibilities suggest themselves: that to be a computer is to be a digital system, for example, or that computation is reducible to an essentially non-sematical intuition underlying the recursion-theoretic notion of effective computability. But since we are focused on the claim that computation is *symbol* manipulation, however, I continue to bar such low-road eliminativist strategies from consideration.

the wings—especially banishing it *qua* semantics—does not even begin to pay the naturalist's debt.

More important is the failure of the empirical reading to satisfy aspect. In the abstract, of course, the reading is so spare that it is hard to know what to say, but its ontological focus is operations, not semantics. But again, maybe this is again cheap: FSM theories of computing are understood to be accounts of *how things work,* not *how they refer*. The real problem, though, is that it isn't clear whether even the aspect that *is* dealt with—i.e., the workings and operations—are explained in a way that matches m·naturalism. Perhaps Haugeland's attempt to provide a definition of (positive) formality that is neither controversial nor semantically-defined (even in opposition) is a step in that direction. The problem with *that*, though, is that, to the extent that he succeeds in meeting the conceptual criterion (semantics-free characterisation), he sidesteps the empirical claim (of whether the resulting *properties* of computers are, empirically, independent of semantics or not).

What about the conceptual version? The case is more complicated, in part because the conceptual reading of "independent of semantics" is not so much a theory of computing as a meta-theoretic condition on the form that theories of computing should take. Although the conceptual reading *itself* fails to satisfy either of m·naturalism or m·reduction, in other words, for many of the same reasons as the empirical reading (it, too, is defined in terms of such unexplained intentional terms as '*theory*' '*syntax*,' '*semantics*,' etc.), in this case this is not a substantive criticism. For it was never the meta-theoretic criterion itself that was supposed to satisfy our naturalistic yearnings. Rather, what was meant to be naturalistically satisfying were the theories on which the meta-theoretic criterion placed conditions. So the question that needs to be asked is not whether conceptual independence *itself* meets the five criteria, but how and whether object-level theories that honour it fare against such standards.

For discussion, let t·syntax and t·semantics be names for theories of syntax and semantics that meet the condition in question—i.e., be such that t·syntax, by hypothesis, is formulable independent of (without reference to) t·semantics. What can be said about the explanatory status of t·syntax and t·semantics?

The first thing to note is that, once t·syntax and t·semantics have been identified and distinguished, and the meta-theoretic conditions governing them laid down, the content of the conceptual reading is exhausted. It has nothing further to say. In particular, the conceptual reading of formality remains silent *on what computation is actually like*. It implies nothing, for example, about whether computation should meet the empirical condition, or even be in any way formal. Having "used up" its take on formality as a meta-level criterion on theoretical form, the vertical theorist must look elsewhere for substantive empirical content. That is why we cannot tell, from Haugeland's definition of formality, whether the machine actually works (empirically) independent of semantics or not (we won't be able to tell until we have in hand an appropriate theory of semantics).

Where things get interesting is at the level of the metatheoretic requirements. Since these explanatory criteria were claimed to be the primary motivation for the conceptual version, it

---

It is reminiscent of children's efforts to achieve independence from their parents by defining themselves in opposition to their parents. As every parent knows, such is not the route to real freedom.

is here where we should expect to see some action. And so indeed we do—though not exactly of the expected kind.

What conceptual independence requires—indeed, its sole purpose—is that t·syntax meet the provisional naturalistic requirement m·naturalism. For a naturalist, this may seem like a glimmer of success (if the reading is true! an issue we have yet to address). But to glimpse is not to have. At least these three considerations would have to be addressed. First, it is so far only a desideratum: all conceptual independence has achieved, in this light, is a claim *that* t·syntax be articulated in semantics-free vocabulary, or at least be dischargeable without recourse to semantical vocabulary; it sheds no light on how that condition might be met. Second, the requirement is strong. Among other things, *it rules out syntax.* Per se, syntax is an intentional notion; you cannot have syntax without there being semantics, at least in the wings. Syntax, that is, is *logically* dependent on semantics (even if it is *ontologically* independent, as the empirical reading suggests). Empirical theorists can perfectly well embrace the notion of syntax as their characterisation of the effective side of the equation; their commitment to the independence of the effective dimension of computation is empirical, after all, not logical. Conceptual theorists, on the other hand, must eschew logical dependence. So they face a demand for a logically-independent-of-semantics characterisation of the effective dimension of computation. Third, is not enough that t·syntax receive a semantics-free rendering; it must be naturalistically palatable throughout. Properly, that is, as mentioned in an earlier footnote, m·naturalism should not have been phrased as requiring that the account be phrased in "language free of undischarged semantical predicates," but in language free of *all* undischarged (or naturalistically unpalatable) predicates. Although it is hard to say exactly what that comes to, in popular theoretic imagination it is taken to imply that the account be reducible to, or at least explanatorily supervenient on, something like the physical substrate. I.e., one must be able to see how the effective dimension is implemented in a naturalistically Sanforized base, that is—or, to put it more familiarly, to see how the *syntax inheres in the physics*.

This is why it is so important to understand the constitution of the cluster of efficacy predicates. Remember, the cluster was initially characterised in terms of eleven putatively allied properties: 'grammatical,' 'syntactic,' 'causal,' 'physical,' 'intrinsic,' 'mechanical,' 'effective,' 'local,' 'potent,' 'shape,' and 'form.' If the work they are collectively asked to do can be reduced to the one physical characterisation, it would be tautological to show that this inheres in the physics, and thus should be easy to satisfy the naturalist's demands. If, on the other hand, the cluster were instead to be resolved in favour of one of the other notions, such as 'syntactic' or 'grammatical,' the chances of showing that it inheres in the physics would seem less good. Finally, there is the possibility that the cluster will not resolve at all, which would almost seem to guarantee reductive failure. And so one way to read this result is simply as putting yet more pressure to come to grips with the nature and status of the potent cluster.

It is instructive to consider a particular case. As we've said, Haugeland is perhaps the clearest and most consistent advocate of the vertical reading. To meet the challenge of providing an non-intentionally characterised t·syntax, he introduces and defends a notion of what he calls an *automatic digital system*—where *automatic* is in turn defined in terms of finite playability, «… fill this out …». So far so good: his characterisation seems at least

ostensibly semantics-free, and thus to satisfy conceptual independence, and thus to satisfy the explanatory criterion m·reduction.

On the other hand, the story is not over; t·syntax is not a theory of computation; as I keep emphasizing, it is a theory of an *aspect* of computation. It is all very well (or at least it is coherent) to dismiss the empirical reading of "independent of semantics" in favour of the conceptual one, and therefore give up any allegiance to the empirical content of the horizontal claim. It is salutary to recognise that doing so requires that one rest the empirical content of the computational theory on a *non*-semantical characterisation of what it is to be formal. And automatic digitality seems at least as good a first proposal as any. Haugeland goes a considerable distance, in other words, towards meeting the (conceptual) reading of formality to which he is committed. But there is a residual requirement. So long as one is working under an overarching formal symbol manipulation construal, what is computational, somehow or other, must still be understood as *symbol manipulation*. For a naturalist, that is, automatic digital systems meet the explanatory criterion m·naturalism, but at too high a price. They achieve the status of a semantics-free account by positing a *semantics-free phenomenon*. And that, of course, will not do. It remains to supply t·semantics.

Haugeland himself is perfectly well aware of this—which is where his own second step comes in. He does not take a computer to be an automatic formal digital system; rather, he takes a computer to be an automatic formal digital system *with an interpretation!* With a single stroke that brings him back into the formal symbol manipulation fold, assuaging this outstanding worry. And by waiting until the last minute to bring in semantics, he is at least arguably able to retain allegiance to the vertical reading. But, at least as regard prospects for naturalisation, he pays a cost for proceeding in this way. For in this final but decisive move, by laying issues of interpretation on the table, the total characterisation (t·syntax + t·semantics) is altered in such a way as to completely violate the explanatory demand m·naturalism. Everything is naturalistically fine *except for the one thing that was naturalistically challenging in the first place*.

4.e. Summary

With this we finally get to the real reason for having pushed so hard on this explanatory phase of the assessment. For the following becomes clear: *if certain ontological conditions were true*, then prospects would radically improve for a naturalistically acceptable resolution to these two outstanding problems for a conceptual version of the "independent of semantics" reading of formality. It will turn out (as we will see in the next section) that these implicitly required conditions are diverse and contradictory. But their influence is enormous. They impose a tacit and steady but confusing background pressure on the attempt to assess, in the open empirical light of day, whether either reading of "formal symbol manipulation" actually holds of practice.

The first issue, about the naturalistic palatability of the syntactic account, puts an *implementation* pressure on the reading of the potent cluster. Reductionist tendencies tend to push one to assume that the confusion about this bunch of properties should automatically be resolved in the "downwards" direction. "Oh, it's not *really* manipulating s-expressions," one is liable to hear someone say; "it's just dealing with ones and zeroes"—or adjusting voltage

levels, or pushing signals around on wires, or whatever. Moreover, this reductionist tendency on the part of theorists lines right up with the constructive orientation of pragmatic computer scientists. If you want to build something, you typically formulate the construction task in terms of lower-level, effective properties of materials out of which the result is to be assembled. Reductionist theorists and practical engineers are metaphysically slated to be the best of friends.

But we mustn't be misled by this fortuitous (or unfortuitous) alignment—that is, we must never confuse the following two things:

a. Reductionist-seeming accounts of how computational systems are implemented; and

b. Genuinely reductive accounts of what computers constitutively are

"It is not really burning; it is just cavorting with oxygen molecules" is wrong, as a description of fire, after all. Even if it *is* cavorting with oxygen molecules, it is still burning. so long as cavorting with oxygen molecules is what burning is,

The semantic problem is more complicated. On the one hand, what would make the reductionist program go through swimmingly would be a similar condition: if the *semantics were implemented in* (or inhered in) *the syntax*. So one should expect pressures for *semantic implementation*, as well—and indeed these are rife throughout informal computational conversations. If semantics were like a Lisp system, and syntax were like a C program implementing that Lisp system, then conceptual independence would hold; and if the C program were physically or naturalistically Kosher, then the Lisp program would be Kosher as well.

Moreover—to bring out one of the genuine ironies in this whole area—*if one understands 'semantics' in the way that computer scientists use the term*, as referring to the program–process relation α, in the figure on page ▪▪, rather than to the process–world relation β, then semantics *would* inhere in the syntax. At least it would inhere in the syntax in the following sense: the semantic interpretation of the program—namely, the behaviour or process to which the program gives rise, upon being executed—would be entirely governed by, reducible to, and informationally specified by the syntax of the program. *That is the whole point of programs.* In fact the informal interest in effectiveness throughout computer science, reflected in its interest in intuitionism and constructive mathematics, turns out from this (mistaken) point of view to exactly match the reductionist desires for semantic implementation. But this is a complete diversion—just one more reason why the ambiguity about programs described in chapter 1 is so theoretically deleterious.

Once one returns to the process–world (β) relation, however, which as we have seen is the relation on which the FSM thesis is founded (at least in the philosophy of mind), it emerges that semantic implementation is in exact conflict with what truth underlies the empirical reading. For starters, semantic implementation is incompatible with what empirical independence means. It is *nonsense* to claim that a C program is independent of the Lisp system it implements, even though it is perfectly coherent (and probably even true) to say that the *account* of the C program is independent of the *account* of the Lisp system it implements—or even, on the logical side (though this is queasier), to say that C program *qua C program* does not involve Lisp types.

Moreover, semantic implementation is *false*. It is even false for logic and arithmetic, as a century's legacy of incompleteness results has taught us. In fact this is surely a, if not the, major result of the whole logical tradition on which the FSM thesis rests: that you cannot capture the substance of a formal system purely in terms of the forms and shapes and effective behaviour of its constituent symbols.

So conceptual independence, backed by naturalistic yearnings, presses for semantic implementation. But semantic implementation is out—false in practice, and recognised as false in the claim of empirical independence. So what does the naturalist do? *Swings completely to the other side.* If semantics cannot be implemented in the naturalistically-acceptable substrate—cannot be treated as a wholly owned subsidiary, as it were—then the second best bet for purity is to be regained by pushing it as far away as possible. This leads to the opposite ontological pressure: the assumption that the semantics of a computation system must be *entirely derived from* (perhaps even attributed from) *the outside*. The second-best naturalistic claim, that is, seems to be completely at odds with the first: a drive towards pure derivative intentionality. "I didn't really mean what I just said about zeroes and ones," the commentator goes on; "computers don't know a thing about them. All they do is push voltage levels up and down. The claim that those voltage levels denote numbers is supplied by us human observers, for our own purposes, completely irrelevant to the workings of the machine *qua* machine." This sentiment is just as common, in informal conversation. And it certainly seems to honour the empirical reading. With a single stroke, though, in spite of its seeming naturalistic motivations, *it robs the account of any chance at all of meeting the naturalistic criterion* m·naturalism. For, as was said at the beginning, an account of semantics, even if only an account of semantic interpretability, must be an essential part of any symbol manipulation analysis of computation that meets the aspect standard.

It is time to let go. We need to look empirically at the empirical reading. It is not enough to point out conceptual confusions and explanatory inadequacies. And it would be premature to try to correct them. A prior question is whether either of these antisemantical claims, conceptual or empirical, is actually *true*.

5. Empirical Critique

The empirical reading fails, in the end, because it is too narrow. The idea of a machine's operating independently of its interpretation is familiar, given reigning myths. And it is plausible, in some very special cases, given semantic disconnection. But neither familiarity nor plausibility can save it. It turns out to hold of only a limited, idiosyncratic species of computation—a class that is not only unrepresentative of full computation in the wild, undermining its empirical significance, but, even worse, that is defined by the very condition that is claimed of it, draining it of conceptual interest.

Fundamentally, the problem stems from the fact that computers, in general and in practice, are *involved* in their subject matters. The FSM thesis, in contrast, is based on an assumption that, in some appropriate causal or metaphysical sense, computers and their subject matters remain *separate*. This alleged separation—a putative gulf between symbolic machine and semantic environment—undergirds not only the antisemantical thesis itself, but the associated analytic framework that takes computation as paradigmatically an activity

consisting of reasoning, representation, inference, or other form of disconnected information processing. What observation shows, however, is that this alleged separation does not obtain in practice. It is the ultimate[21] impossibility of wholly pulling words and worlds apart that defeats the formalist claim.

5.a Diagnosis

Many counter-examples to the FSM thesis can be cited. Time pressing, I will cut straight to the heart of the matter by highlighting a distinction between two kinds of "boundary" thought to be relevant or essential—indeed, often assumed *a priori*—in the analysis of computers and other intentional systems:

1. A physical boundary, between the system and its surrounding environment—i.e., a distinction between "inside" and "outside"; and
2. A semantic boundary, between symbols and their referents.

In terms of these two distinctions, the empirical version of (the "independent of semantics" reading of) the formal symbol manipulation construal can be understood as presuming the following two proposition:

1. alignment: An assumption that the physical and semantic boundaries line up, so that all the symbols end up inside, all the referents outside; and
2. isolation: A claim that this putatively aligned boundary is a barrier[22] or gulf across which dependence (causal, logical, explanatory) does not reach.

The fundamental idea underlying the formal symbol manipulation thesis, that is, is that a barrier of this double, allegedly-aligned sort can be drawn around a computer, separating a pristine inner world of symbols—a private kingdom of thought or reasoning, as it were—seen both to work (ontologically) and to be analysable (theoretically) in relative isolation, without distracting influence from the messy, unpredictable exterior.

Note, first—and not surprisingly—that the traditional "intuition pumps" and paradigmatic examples in terms of which the fsm construal is motivated, such as theorem proving and numerical computation, meet this complex condition. First, they involve internal symbols designating external situations, thereby satisfying alignment: data bases representing

---

'It is the 'wholly' that is the villain. In the eventual positive story, word and world will get pulled apart *to some extent.* But not completely.

The ambiguity about the type of dependence reflects much of the ambiguity in published discussion. Thus what we have here called the conceptual version of "independent of semantics" rests on something like an assumption of *logical* independence; the empirical version, on *causal* independence, in which the putatively aligned boundary is thought to be a barrier to the flow of causation or effect. Except that it is of course not so simple, because everyone recognizes that computers need sensors and effectors. For now, however, I d not want to get mired in the details of specific readings, so much as to paint a simple, overarching picture. For it is the picture in the large—the alleged existence of a single moat, *barrier to dependence of all relevant sorts*—that, I believe, underwrites intellectual history's allegiance to the "independent of semantics" idea. Or, to turn it up the other way, the ontological condition in the world that would have to hold, in order for a generally "independent of semantics" claim to be maintained as a constitutive condition of computation, would be for this overarching divide to be preserved in some form at all these various different levels.

employee salaries, differential equations modeling the perihelion of Mercury, first order axioms designating Platonic numbers or purely abstract sets. Second, especially in the paradigmatic examples of formal axiomatizations of arithmetic and proof systems of first-order logic (and, even more especially, when those systems are understood in classical, especially model-theoretic, guise), the system is assumed to exhibit the requisite lack of interaction between the syntactic proof system and the model-theoretic interpretation, satisfying isolation. In conjunction, the two assumptions allow the familiar two-part picture of a formal system to be held: a locally contained syntactic system, on the one hand, consisting of symbols or formulae in close causal intimacy with a proof-theoretic inference regimen; and a remote realm of numbers or sets or "ur-elements," in which the symbols or formulae are interpreted, on the other. It is because the formality condition relies on both theses, that is, that the classical picture takes computation to consist exclusively of symbol–symbol transformations, carried on entirely within the confines of a machine.

5.b Cross-cutting boundaries

The first—and easier—challenge to the antisemantical thesis comes when one retains the first alignment assumption, of coincident boundaries, but relaxes the second isolation claim, of no interaction. This is the classical realm of input/output, home of the familiar notion of a transducer. And it is here that one encounters the most familiar challenges to the fsm construal (such as the "robotic" reply to Searle's (1980) Chinese room argument, and Harnad's (1991) "Total Turing Test" as a measure of intelligence). Thus imagine a traditional perception system—for example one that on encounter with a mountain lion constructs a symbolic representation of the form mountain-lion-043. There is interaction (and dependence) from external world to internal representation. By the same token, an actuator system, such as one that would allow a robot to respond to a symbol of the form cross-the-street by moving from one side of the road to the other, violates the independence assumption in the other direction, from internal representation to external world.

Note, in spite of this interaction, and the consequent violation of isolation, that alignment is still preserved in both cases: the transducer is imagined to mediate between an internal symbol and an external referent. Yet the violation of isolation alone is enough to defeat the formality condition. This is why transducers and computation are widely recognized to be uneasy bedfellows, at least when formality is at issue. It is also why, if one rests the critique at this point, defenders of the antisemantical construal are tempted to wonder, given that the operations of transducers violate *formality*, whether they should perhaps be counted as *not being computational*.[22] Given the increasing role of environmental interaction within computational practice, it is not at all clear that this would be possible, without violating the condition of empirical adequacy embraced at the outset. But it does not matter, ultimately, because the critique is only half way done.

. Thus Devitt, in (1991), restricts the computational thesis to what he calls "thought-thought" (t-t) transactions; for him output (t–o) and input (i–t) count as non-computational.

5.c Participatory involvement

More devastating to the fsm construal are examples that challenge the alignment thesis. On analysis, it turns out that far from lining up on top of each other, real-world computer systems' physical and semantic boundaries *cross-cut*, in rich and productive interplay. It is not just that computers are involved in an engaged, participatory way with *external* subject matters, in other words, as some recent "situated" movements have suggested. They are participatorily engaged in the world *as a whole*—in a world that indiscriminately includes themselves, their own internal states and processes. This integrated participatory involvement, furthermore, blind to any *a priori* subject-world distinction, and concomitantly intentionally directed towards both internally and externally exemplified states of affairs, is not only architecturally essential, but is also critical, when the time comes, in establishing and grounding a system's intentional capacities.

From a purely structural point of view, four types of case are required to demonstrate this non-alignment: (i) where a symbol and referent are both internal; (ii) where a symbol is internal and its referent external; (iii) where symbol and referent are both external; and (iv) where symbol is external and referent internal.

The first case—where both symbol and referent are internal—is exemplified throughout computation: in cases of quotation, meta-structural designation, window systems, e-mail, compilers, loaders, network routers, and at least arguably all programs (as opposed, say, to databases). Elementary counting may be the simplest example. Suppose that a computer is programmed to count the number of characters in a string. Given "abc", that is, it would return '3'. Now '3' is not a *number*, of course; it is a *numeral*. As we said above, numeric computing is *numeral* crunching, not number crunching, But what does the number denote? It denotes the *number of elements in the list*. And that number

…

The second, of internal symbols with external referents, can be considered as something of a theoretical (though not necessarily practical) default, as for example when one remembers the sun's setting over a lake. The third and fourth are neither more nor less than a description of ordinary written text, public writing, etc.—to say nothing of pictures, sketches, conversations, and the whole panoply of other forms of external representation. Relative to any particular system, they are distinguished by whether the subject matters of those external representations are similarly external, or are internal. The familiar red skull-and-cross-bones signifying radioactivity is external to both man and machine, and also denotes something external to man and machine, and thus belongs to the third category. To a computer or person involved, on the other hand, an account of how they work (psychoanalysis of person or machine, as it were) is an example of the fourth.

By itself, violating alignment is not enough to defeat formality. What it does accomplish, however, is to radically undermine isolation's plausibility. In particular, the antisemantical thesis was challenged not only because these examples show that the physical and semantic boundaries cross-cut, thereby undermining the alignment assumption, but because they illustrated the presence, indeed the prevalence, of effective traffic across both boundaries—between and among all the various categories in question—thereby negating isolation.

And this, in turn, shows up, for what it is, the common suggestion that transducers, because of violating the antisemantical thesis, should be ruled "out of court"—i.e., taken not to be computational (à la Devitt (1991)). It should be clear that this maneuver is ill-advised; even a bit of a cop-out. For consider what a proponent of such a move must face up to, when confronted with boundary non-alignment. *The notion of a transducer must be split in two.* I.e., someone interested in transducers would have to distinguish:

1. *Physical transducers*, for operations or modules that cross between the inside and outside of a system;
2. *Semantic transducers*, for operations or modules that mediate between symbols and their referents.

And it is this bifurcation, finally, that irrevocably defeats the antisemantical claim. For the only remotely plausible notion of transducer, in practice, is the physical one. That is what we think of when we imagine vision, touch, smell, articulation, wheels, muscles, and the like: systems that mediate between the internals of a system and the "outside" world. Transducers, that is, at least in informal imagination of practitioners, are for connecting systems to their (physical) environments.[24] What poses a challenge to the formal (antisemantical) symbol manipulation construal of computation, on the other hand, is the *semantic* transducer: those aspects of a system that involve trading between an occurrent state of affairs, on the one hand, and a representation of it, on the other. Antisemantics is challenged as much by disquotation as by driving around.

As a result, the only way to retain the empirical version of the fsm construal of computation is to disallow (i.e., count as non-computational) the operations of semantic transducers. *But that is absurd!* It makes it clear, ultimately, that distinguishing that subset of computation that satisfies the empirical version of the antisemantical claim is not only unmotivated, solving the problem by fiat (making it uninteresting), but is a spectacularly infeasible way to draw and quarter any actual, real-life, system. For no one who has ever built a computational system has ever found any reason to bracket reference-crossing operations, or to treat them as a distinct type. Not only that; think of how many different kinds of examples of semantic transducers one can imagine: counting, array indexing, e-mail, disquotation, error-correction circuits, linkers, loaders, simple instructions, data base access routines, pointers, reflection principles in logic, index operations into matrices, most Lisp primitives, and the like. Furthermore, to *define* a species of transducer in this semantical way, and then to remove them from consideration as not being genuinely computational, would make computation (minus the transducers) antisemantical *tautologically*. It would no longer be an interesting claim on the world that computation was antisemantical—an insight into how things are. Instead, the word 'computation' would simply be shorthand for antisemantical symbol manipulation. The question would be whether anything interesting was in this named class —

---

[24] . This statement must be understood within the context of cognitive science and the philosophy of mind. It is telling that the term 'transducer' is used completely differently in engineering and biology (its natural home): to signify mechanisms that mediate changes in *medium*, not that cross the inside/outside *or* symbol/referent boundary.

and, in particular, whether this conception of computation captured the essential regularities underlying practice. And we have already seen the answer to that: it is *no*. In sum, introducing a notion of a semantical transducer solves the problem tautologically, cuts the subject matter at an unnatural joint, and fails to reconstruct practice. That is quite a lot to have against it.

Furthermore, to up the ante on the whole investigation, not only are these cases of "semantic transduction" all perfectly well-behaved; they even seem, intuitively, to be as "formal" as any other kind of operation. If that is so, then those systems either are not formal, after all, *or else the word 'formal' has never meant independence of syntax and semantics in the way that the* fsm *claim construes it*. Either way, the empirical version does not survive.

…

Here is where we stand. Both theses that sustain the empirical version of the antisemantical thesis are false. alignment is false: physical and semantic boundaries cross-cut in a myriad ways. And isolation is false: in practice, neither boundary is a causal or effective, let alone a logical, moat. As a result, the only way one could save the empirical reading of the thesis would be to impose it in advance, by fiat, and hide under the notion of transduction everything that did not fit. But to do that would be to select out an untenably narrow, unmotivated subset of practice, bearing no relation to the class of phenomena that the word 'computation' is actually used for. Not only that; on reflection, the empirical version does not even bear a very straightforward relation to those phenomena (computational or not) that are traditionally counted as formal—witness the prevalence of quotation, disquotation, reflection principles, etc., in ordinary formal logic. And so the "independent of semantic" reading must go. All that remains is its tautological shadow.

…

6. Participation

Criticism only goes so far. It is time to start constructing a positive alternative.

What have we learned? A number of things:

First, computers participate. They are *involved* in their subject matters. Second, sign and signified may sometimes be separated, but they are not always separated. In general, semantic separation will be partial. Third, the semantic boundary, such as it is, between symbol and referent, cross-cuts the physical boundary, between inside and out. This cross-cutting of boundaries was taken to be a general characteristic of participatory intentional systems.[25] Fourth, although we cannot be entirely sure until we have an accepted semantical theory, it looks as if the empirical independence condition—Fodor's original formality condition—is wrong. The way computers work, mechanically, seems to be neither empirically nor conceptually independent of their semantics.

That last point must not be overinterpreted. In spite of that lack of independence, nothing has been said to imply that semantical properties are potent or effective—*ever*. Intuitive considerations at the outset (the remoteness of Pluto, the nature of hypothetical reference)

---

[25] «Write a note about how this will reconstruct what has gone on under the label "situated"»

suggested that semantical properties are not *always* effective. It is entirely compatible with all the evidence we've amassed that they are *never* effective. For all we know, that is, it may be that computers *never* work in virtue of the exemplification of semantical properties—even if they don't work independently of the, either.

Is this a contradiction? No. Consider three theses:

i1. Computational symbols are manipulated *independent of* their semantic properties.

i2. Computational symbols are manipulated *in virtue of* their semantic properties.

i3. Computational symbols are *not* manipulated in virtue of their semantic properties.

The first, i1, is Fodor's formality condition. But i2 is not its negation. It is i2 and i3 that are opposites: computers do, or don't, work in virtue of the exemplification of semantical properties. That is not to say that either of them need be true. It could be—might well be, for example, in cases where disconnection or separation is lacking—that computers *sometimes* work in virtue of the exemplification of semantical properties. And yet, nothing that has been said requires us to believe such a thing. Moreover, as it happens, I don't believe it. For very non-standard metaphysical reasons I happen to believe i3.

Of primary interest here, however, is the difference between i1 and i2. It sometimes seems thought that if two things are not the same, then they must be independent. But no such conclusion follows. In fact there is surely a vast realm between identity and independence: a realm of *partial interdependence*. Take a folksy example: are we dependent on, or independent of, our parents? Likely neither. Psychologically, in fact, complete dependence and complete independence are considered *pathological*. Why shouldn't they be pathological in other cases, as well? Perhaps they are pathological in cases of participation. At any rate, this much at least seems true: with respect to computation in the wild, it looks as if syntax and semantics are partially interdependent—perhaps even partially co-constitutive.

But we were looking for a positive picture. What image of computation is compatible with these insights?

Think of it this way. In any given situation, vast numbers of things are the case. All sorts of objects exemplify all sorts of properties and stand in all sorts of relations. Infinitely many, in fact—without batting an eyelash. Just think of how many relations my mouse and keyboard stand in: of being less than one foot apart, of being less than two feet apart, of being less than three feet apart, and son on. Ontological promiscuity may (or may not) be dubious at the level of the types, but at the level of the tokens, promiscuity is the order of the day.

Suppose you want to create an intelligent creature—that is, a creature that would act appropriately (i.e., produce behavior τ) in all sorts of states of affairs σ. Conceptually, the simplest approach would be to build a mechanism that responded directly to σ—i.e., that mediated the relation $\sigma \rightarrow \tau$, for each relevant σ and τ.

But there is a problem. In a large number of cases, the "triggering" state of affairs σ *won't be causally effective*. That is, σ will not be the sort of event such that $\sigma \rightarrow \tau$ is causally available. Or, to put the same point another way, there do not exist nomological relations tying the presence or absence of σ to *any* appropriate (local and available) causal states. *Some* things *will* be local and effective, such as (perhaps) whether you are about to crash into

a wall. It is just that the vast majority of things you'ld like to respond to don't "make themselves available" as causal inputs.

What to do? Well, the story is obvious, and not even unfamiliar: you set up states that *are* causally effective to stand in for those that are *not* effective. Fortunately, the world comes with a lot of local degrees of freedom (what in another context I have called "slop"): your creature can by and large adjust its internal causal state—wiggle and stir and flip a few bits—without wreaking havoc on the sustaining physical surround. So (to gloss a hundred million years of evolution and an equal number of conceptual issues), you figure out how to have locally available states that *are* effective stand in for all the states you are interested that are *not*.

Isn't this the standard picture—the picture we just spent so much time dismantling? No, it is not; and for an interesting reason. It's not wholly opposed to the standard picture—but that's okay; a lot was right about the standard picture. It just differs on all the crucial details.

Four details, for starters.

First, what matters, ultimately, does not have primarily to do with what is local and what is distal (or with what is inside and what is out). Rather, what matters is whether it can do any *work* for you (in that same, mechanical or effective sense of "work").[20] If it is outside, even a long way away, but still causally available, then use it directly. If it is not causally available, then set up some structure—some sign or symbol or data structure—to serve in its stead. Or (a third option) get yourself driven by some other state of affairs σ' that *is* causally effective and that *correlates* with σ, at least in the range of circumstances in which you hope to survive or be effective. Or (here comes inference) follow a path through a sequence of effective states that you have reason to believe will end you up with such a state of affairs σ'. Sometimes one strategy will work better; sometimes another. The point is just that if you don't do *one* of those things, you won't be able to track the original situation your are interested in—presumably at your peril.

Second, computation, as we've seen, is participatory. The effective physical dimension (the realm of the causally efficacious) is one dimension of the reality in which computation is embedded. What the strategy just outlined comes to is this: the critter must "project" a bunch of the non-effective things it *is* interested in onto an available range of the effective dimension that it are not *otherwise* interested in. Fortunately, as we've already said, there is generally a lot of "unused bandwidth" within the realm of the effective. Most medium-scale objects (like brains and CPUs) contain an almost infinite amount of it. Re-arrange a few molecules here and a few molecules there, and you can produce bewilderingly complex circuits that occupy very little space (on the order of a million transistors per square millimeter of integrated circuit, these days). No, finding available effective structures to arrange in this clever ways is not, usually, the problem. There are, however, limits as to what you can *do* with these arbitrary arrangements. Radical limits, as it happens. Given any one

---

This is what is right about the "use the world as its own model" intuition that permeates so much recent embedded and neo-robotic AI. That's a great idea—just so long as the aspect of the world that you are interested in can *drive* you, can be exploited to do effective work. If I am interested in knowing whether my belief are consistent, or whether anything within 10 miles of me is likely to be mentioned in today's *New York Times*, I can't "use the world as its own model" because those states of affairs σ *can't cause anything in me*. I can't, so to speak, *touch* or *see* them.

configuration of effective material, it turns out that substantial space, time, and energy considerations limit how fast and how efficiently you can get it to transmogrify into another effective state, even if the *path* from one to the other is well-defined.

And with this, of course, as you may have guessed, we land squarely into the recursion-theoretic or Turing machine conception of computing. Remember what I said about "stuff manipulation"? Well, the theory of computing at large within theoretical computer science—computability theory, complexity theory, the analysis of algorithms, etc.—is exactly that: a theory of stuff manipulation, currently digital stuff, but rapidly being expanded into continuous stuff as well. "Stuff" doesn't mean *arbitrary* stuff, including angels and set-theoretic structures and pure types. Rather, "stuff," in this context, is the causal or effective dimension of the world. The mathematical theory of computability, in other words—will eventually be recognised not to be a theory of anything intentional at all (including: not being a theory of computing). Rather, it is a *mathematical theory of causality*—at least of those aspects of the intuitive notion of causality that will succumb to scientific analysis. To avoid confusion, I call it a "mathematical theory of effectiveness" (or a theory of the "flow of effect"). Because, superficial evidence notwithstanding, it is utterly unconcerned with intentional issues, it is much closer in both style and substance to *physics* than it is to its forebear, logic. In fact I will go out on a limb: within 50 or 100 years, I predict, this body of mathematical work—this theory that is *called* "The theory of computation," but that *isn't* a theory of computation at all, if I am right—will have merged with dynamics and the rest of physics, in something of a crowning achievement of the natural sciences.

A theory of effectiveness is not a theory of computing, because effectiveness is merely a scientific reconstruction of the bumping and shoving of the world, whereas computing is a strategy for *exploiting* the bumping and shoving of the world so as to compensate for what is *not* effective—and thereby triangulate in on a much larger fragment of the world. But a theory of how to "compensate for what is not effective" is nothing less than a theory of semantics—and so is *not* a part of the natural sciences, at least not anything like the natural sciences that have reigned supreme in the last 300 years, but instead part of what we might call the *intentional* sciences, which, to hazard another prediction, I predict will reign supreme for the next 300 years.

Except that that gets it backwards. It is not that a theory of how to "compensate for what is not effective" is a theory of semantics—as if we knew what semantics was, and propose to use that to explain what it was to compensate for what is not effective. The naturalistic reading goes the other way: what a theory of semantics is a theory of is how creatures—machines, people—compensate, within the realm of what is effectively available, for things that are not effectively available but that are nevertheless important to them. The world as a whole is a (continuous) mixture of effective and non-effective, of course, so this is not one realm set up neatly and completely to represent or correspond to another. Rather, as the

---

The argument that computability theory, complexity theory, the theory of Turing machines, and the like, doesn't have to do with symbols or interpreted phenomena, but is rather (in spite of the protestations of its practicioners) what is claimed in the text: a mathematical theory of effectiveness, is given in *The Age of Significance, Volume III: Effective Computability* (forthcoming).

participatory nature of things makes clear, it is an expedient and often haphazard exploitation of whatever means are available so as to help one deal with things that are not.

7. Conclusion

What about narrow and broad content? What about elms and experts? What about indexicality, situatedness, and asymmetrical dependency?

This is the third part of the paper, which Fodor will present, in his comments. I'm not sure what he will say, though I am sure it will be right. But since we're not quite ready for him yet, let me make a few predictions.

First, broad and narrow content are vulnerable to the failure of alignment.

…

…

…

—— *end of file* ——