 Last edited Tuesday, March 6, 2001

A Theory of Marks (cont'd)

I. Review

A. Review

1. Last time, I argued that the "encoding" relation (ρ) between the marks (numerals) on the tape of a Turing machine and the numbers that those marks supposedly denote or represent—specifically in the case of the marks on the output tape of a machine that attempts to solve the halting problem, but by implication more generally as well—is the *inverse* of what is normally assumed:
   a. The numbers represent (model) the marks
   b. Rather than the usual assumption: that the marks represent (encode) the numbers.
2. What this led us towards was a view of computability theory that made it look like other mathematical theories in science: as an enterprise that *indirectly classifies* one set of (typically physical) phenomena in terms of various other (abstract) mathematical entities—in the centuries-long (Cartesian) tradition of the arithmetisation of science.

B. Causality

1. If this analysis is right, then the traditional (official) "theory of computation"—complexity theory, the theory of computability, recursion theory, etc.—has *nothing to do with representation, semantics, or intentionality*.
2. It also has nothing to do (at least nothing *obvious* to do) with computing *functions. values, numbers, etc.* It cannot have to do with them, because the only way to understand the behaviour of a machine that manipulates marks *as* computing a number or function (and it is only at the level of the marks, and transformations on them, as we have argued, where computing "happens") is by *interpreting* the marks as denoting numbers. However, as we have seen, this is not how the marks are being used. Rather, the numbers are used to classify the marks. As a result, *the "computing a function" (and "deciding a question") interpretation of Turing machine behaviour must go.*
3. That is: the "uses" of semantic (encoding) relations are not (ontological) parts of the genuine subject matter, but rather are (epistemological) parts of the theoretical machinery used to understand it.
4. And yet, in spite of setting these "interpretations of the marks" aside, we are still assuming that computation itself *is* (remains) an intentional / semantic phenomenon.[1]
5. In order to make the grade, therefore—as we have said since the beginning—a comprehensive theory of computation will have to treat computing's semantical aspect.
6. It follows that we can immediately conclude the following:

---

[1] Cf. the "major dialectic"—the mind/body problem for machines, or the interplay of *meaning* and *mechanism*.

◆ The official theory of computation is not in fact a theory of computation after all.

7. Rather, what we have determined that the EC construal is is a **theory of marks**: a theory of the bumping and shoving of the world, of the organisation and consequences of the effective aspects of states. That is:

P6 In spite of the press, the traditional (official) theory of "effective computability" *is not a theory of computation at all.* Rather, it is a *mathematical theory of the flow of effect*—that is, a 20th-century **mathematical theory of causality**.

C. Status
   1. **P6** is of course a very strong claim—with which many will disagree.
   2. Before dismissing it, though, note that it is not an *indictment* of the theory. That is: we are not claiming that the labours of those who developed this theory were invested in vain.
   3. On the contrary, a theory of *what can be done effectively (mechanically) in this world*—and of how hard it is to do it—would be an enormous achievement
   4. In fact we can say something stronger:

P7 Our reconstruction of the theory of computability as a mathematical theory of the flow of effect—i.e., a mathematical theory of causality, a theory of the powers and limitations of the material "body"—makes it *more* relevant to a comprehensive theory of computation (because it applies to *all* of computing) than the official formulation, which claims it to be a theory of: calculating the values of functions at argument positions (which will always remain a specialized case).

II. Status

A. Where do we stand with respect to claim **P7**?
B. We have identified several reasons to believe it.
   1. We were driven to it in order to make sense of why deviant machine ★ was a cheat.
   2. More generally, we came to it in the process of tying down adequate constraints on the notion of a "reasonable encoding"
   3. More seriously, we saw that understanding things in this way actually *explains* a number of facts we had observed along the way about "reasonable encodings"
      a. **Simplicity**
         i. *Condition:* Why ρ has to be so simple—for example, containing no equality.
         ii. *Explanation:* because it has to be invertible. In fact this reconstruction makes the simplicity (and mathematisation) maximally clear. It is no wonder that the (theoretic) ρ relation is massively simpler than generalized interpretation functions defined over actual computational states. Equality, for example, is barred because its inverse is *ambiguity*, which is not something one wants in a theoretic model.[2]

[2]See figures 3a and 3b in lecture 9a (page 9·6).

b. **Invisibility**
   i. *Condition:* Why ρ is typically not theorised as such
   ii. *Explanation:* because it is part of the theoretician's ordinary mathematical tools and equipment, not part of the primary subject matter. In general, sciences are bound to provide a meta-theory of their theoretical tools and equipment.

c. **Ubiquity**
   i. *Condition:* Why there is informal theoretic consensus on the nature of ρ, which in practice seems adequate
   ii. *Explanation:* same as above: this is to be expected in any mathematical science.

d. **Theoretic irrelevance**
   i. *Condition:* Why, in spite of the crucial (and constrained) role we saw that ρ plays, there is a strong intuition among practitioners that the nature of ρ is not critical to the main theoretic results.
   ii. *Explanation:* Ultimately, ρ is *always* substantially irrelevant—because it is part of the theoretical equipment. What is relevant is *what* the numbers and functions are a *mathematical model of*.

4. Because of these successes of the (proposed) reconstruction, I will assume that anyone who wants to argue against it will need to produce a different analysis of the (evidently crucial) notion of a reasonable encoding, that explains **(at least) the foregoing four facts.**

C. In addition, I am prepared to (and in *AOS Volume III* do) argue that it makes sense of a number of other things:

> 1. Why complexity results (e.g., that some algorithm is logarithmic, or cubic, or whatever) are defined with respect to the *lengths of the inputs* (taken as marks), rather than with respect to the *magnitude of the argument* to the function;
> 2. The recent emergence of Girard's linear logic, intuitionistic type theories, programming language semantics, etc.
> 3. The popular use of term models, etc., in analysing the semantics of programming languages (such as Prolog);
> 4. The fusion of the theory of computing with quantum mechanics (e.g., tying together of the loss of a bit of "information" and the production of a minimal quantity of heat), discussed as a foundational unification project in *AOS* volume I.

D. If we had time, we would spend several weeks in this course going over all of these things

E. We can also identify some reasons why a theory of this sort would be *good*
   1. It could give us a leg up on a theory of "body"
   2. It discharges (positive) half of the FSM claim (see below)
   3. More broadly: a 20th century theory of causality
      a. Makes sense of concern with *materiality*.
      b. Cf. discussions in surrounding disciplines about physicality, causal theories, etc.

F. Consequences
   1. Nevertheless, the consequences of this reconceptualisation are **dire**
   2. Among other things, it means that we will have to reconstruct all the relative computability (complexity) results, such as prime factorization
   3. Will also have to reconstruct absolute computability results
   4. Including the unsolvability of the halting problem!
   5. The reason that these have to be reformulated is that, as things stand, they are framed *in terms of the lowest (denoted, mathematical) realm*, in the "three-storey" picture. But if our analysis is right, then that way of framing them is not just infelicitous; it is actually *wrong*.
   6. Instead, will have to be framed in terms of structural relationships between
      a. The (effective) structure of a machine
      b. The (effective) structure of its inputs
   7. Turn to some of these next time.
   8. For now, the main thing to keep in mind is this:

> **P8** All we have argued (so far) is that there are good reasons to support a (causal) reconstruction of the theory of effective computability. We have not yet *done* that reconstruction. That works remains.

G. Today, rather than setting out on this reconstruction, I want to explore what was just mentioned in II.D: how the proposed reconstruction, if it were done, would fit into our overall project, of developing a comprehensive theory of computing—one that meets our four (three formal and one substantive) criteria of adequacy.

III. Computation

A. What does the proposed reconstruction have to do with computing?
   1. Since the outset, characterized computers in terms of a fundamental dialectic between:
      a. "Mind": the semantic or **meaning** aspect; and
      b. "Body": the physical, effective, or **mechanism** aspect
   2. The proposed reconstruction suggests that the mathematical theory of effect:
      a. Gives us a leg up on the second half of this: the theory of "body"
      b. As such, discharges (positive) half of the FSM claim
      c. Makes sense of enduring concerns with **materiality**
         i. Cf. discussions in surrounding disciplines about physicality, causal theories, etc.
   3. But the semantic side remains
      a. Constitutive of real computing (computation-in-the-wild)
      b. Ripe for explanation

B. Irony
   1. Note that there is something of an irony in how this has worked out.
   2. As we saw (and emphasized at the end of Part II), one reading of the FSM claim—what we called the "conceptual" sub-reading of the negative reading of formality—was that one could *give an account of computing* that was free of any reference to semantics.

3. What the EC construal has accomplished, thus, is what (that reading of) the FSM construal promised: a semantics-free characterisation.
4. It is semantics-free (at the level of explanation) because it merely adverts to the causal structure of the world.
5. But as we know (from the FSM analysis), just because it refers to the causal structure of the computer, that doesn't mean that it necessarily doesn't talk about the semantic aspect.
6. We should ask, therefore, whether the EC construal is (ontologically) semantics free—in the sense of not providing (even a reductive) account of computing?
7. The answer to that question depends on what semantics is like.
   a. Remember the participatory morals of the first critique.
   b. If it turns out that semantics *can in some instances be causal* (as many people think), then we cannot conclude, from the fact that an EC account is semantics-free (at the level of explanation), that it is an account of semantics-free phenomena.
8. However, if (as we hypothesized at the end of the first critique, in Part II) semantic properties are intrinsically *non-effective*, then it would follow that the EC construal is ontologically semantics-free as well.
9. However to determine that awaits a theory of semantics—which we don't yet have.

C. Summary

1. These various results can be understood in terms of the series of diagrams given in figure 1 (on the next page).
   a. The vertical dimension represents the dimensions or aspects of things, with the effective or mechanical at the top, the semantic or interpretive at the bottom
   b. The horizontal dimension is simply the set of all possible devices
2. Note: It remains an open question whether to be a computer is simply to be a device with a semantic or interpretive aspect (as the diagrams suggest), or is something narrower (in which case the horizontal span of "computers" should be narrower than indicated).
3. In these terms, we can distinguish four possible (candidate) theories:
   a. A comprehensive theory of computing — This is what we are after (fig 1a)
   b. A general theory of mechanism — A theory of causality (fig 1b)
   c. A theory of digital mechanisms — I.e., (1b) restricted to discrete case (fig 1c)
   d. A theory of computing functions — Computation, restricted to mathematics
4. Remarks
   a. The current proposal to understand the Turing-theoretic theory of computability (recursion theory, complexity theory, etc.) as a theory of the causal structure of digital mechanisms is as reflected in diagram (1c)
   b. The suggestion that one could interpret current computability theory as a theory of computing functions requires identifying diagrams (1c) and (1d).
      i. It would require assuming that the representation relation, ρ, from tapes to FAVs (functions, arguments, and values) be *the same as* the (untheorised) modeling relation used by the theorist to classify effective mechanisms.
      ii. Even if this identification could be shown to hold, the resulting theory would at best *identify* the class of such computing machines; it would not *explain* their semantic as-

pect—because ρ, rather than being explained directly, is instead piggy-backed off the (untheorised) modeling relation.

D. Prospects
   1. If this analysis is right, considerable work remains to be done
   2. Computability and complexity

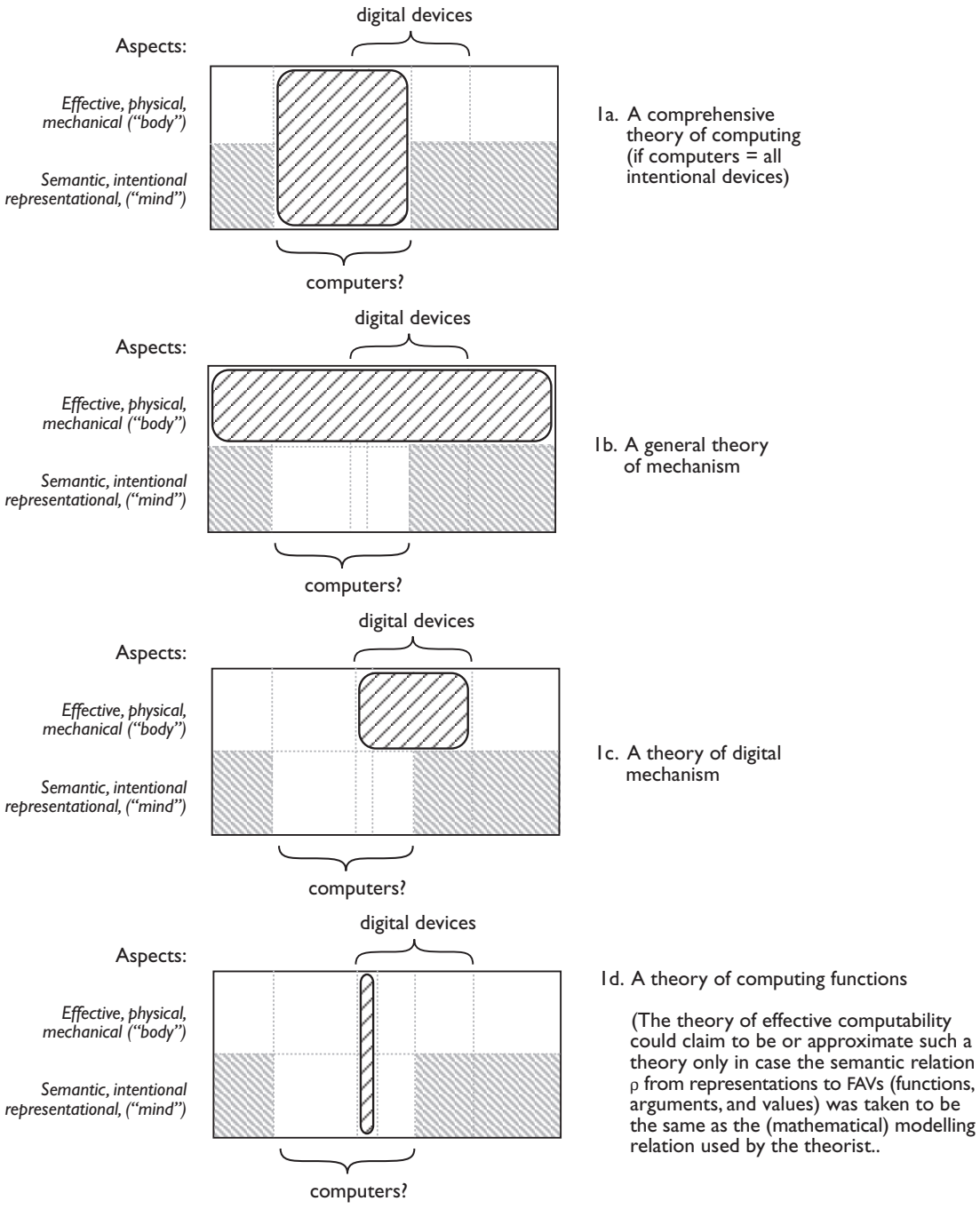

Figure 1 — Coverage of various possible theories of computing

   a. All extant results (of absolute computability, complexity, etc.) will have to be reframed *without reference to the interpretation of tapes*
   b. Some of this is already implicit in practice—e.g., the fact that the complexity of number-theoretic algorithms (whether they are logarithmic, cubic, exponential, or whatever) is defined with respect to the *lengths of the inputs* (taken as marks), rather than with respect to the *magnitude of the argument* to the function.
   c. Also: we will need to reinterpret the halting problem: formulation, result, and what is intrinsically going on—not in terms of other (modeled) machines, but in terms of structural relationships between:
      i. The (effective) structure of a machine
      ii. The (effective) structure of its inputs
3. Physicality
   a. We have *claimed* that the distinctions between different mark types, etc., are effective
   b. If they are indeed being implicitly defined with respect to a background physicalist intuition, then the connection with physics will need to be spelled out explicitly
      i. This is the project that Gandy started in his "Principles of Mechanism" paper
      ii. One would also want to include the work of Landauer et al in fusing computation theory and quantum mechanics (as discussed in *AOS volume I chapter 1*).
   c. More generally, it is possible that this ("foundational") branch of computer science should be merged with physics
      i. Very interesting questions of scale to be addressed: what is it for the results to be framed at *arbitrary levels of physical scale*
      ii. Also questions about the *ontological warrant* for non-basic physical entities (e.g., machine states, mark types, etc.) in terms of which the results apply.
      iii. Questions remain, such as: Can these be identified or individuated without implicit reference to semantics?
4. Review the rest of theoretical computer science
   a. For example, as suggested above, I claim that this perspective makes more sense of practice (e.g., programming language semantics, the emergence of Girard's linear logic, intuitionistic type theories, the π-calculus, etc.) than the standard one
   b. We'll look at some of these things in the coming classes during the rest of the semester.
5. Next week, though, we will instead look at something that deepens our understanding of how the notion of a *program* arose—and also at the emergence of two different kinds of semantic relation—what in an earlier lecture (and in the "100 Billion Lines of C++" paper available on the class web site) I called **program semantics** and **process semantics**.

—— *end of file* ——