The Rise of Programs

I. Review

A. Intro

1. Plan: talk today about **programs**—where they came from, how to understand them, etc
2. But first a bit of review

B. Where are we? We are in the midst of the analysis and critique of the second construal or "effective computability" (EC) construal, that computing is whatever it is that underwrites the notion of Turing machines, the official "theory of computation," "theory of (effective) computability"

C. Status

1. What we argued is that, in spite of its surface, that official body of theory is a not a theory of computing at all, because it doesn't deal with the fundamental dialectic, which we have claimed underwrites computation in the wild: the interplay of mechanism *and meaning*.
2. In particular, the theory didn't deal with computing's semantic or "meaning" aspect
3. Instead, we diagnosed the EC construal as 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.
4. That is (as we said in lecture notes 9b):

> **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**.

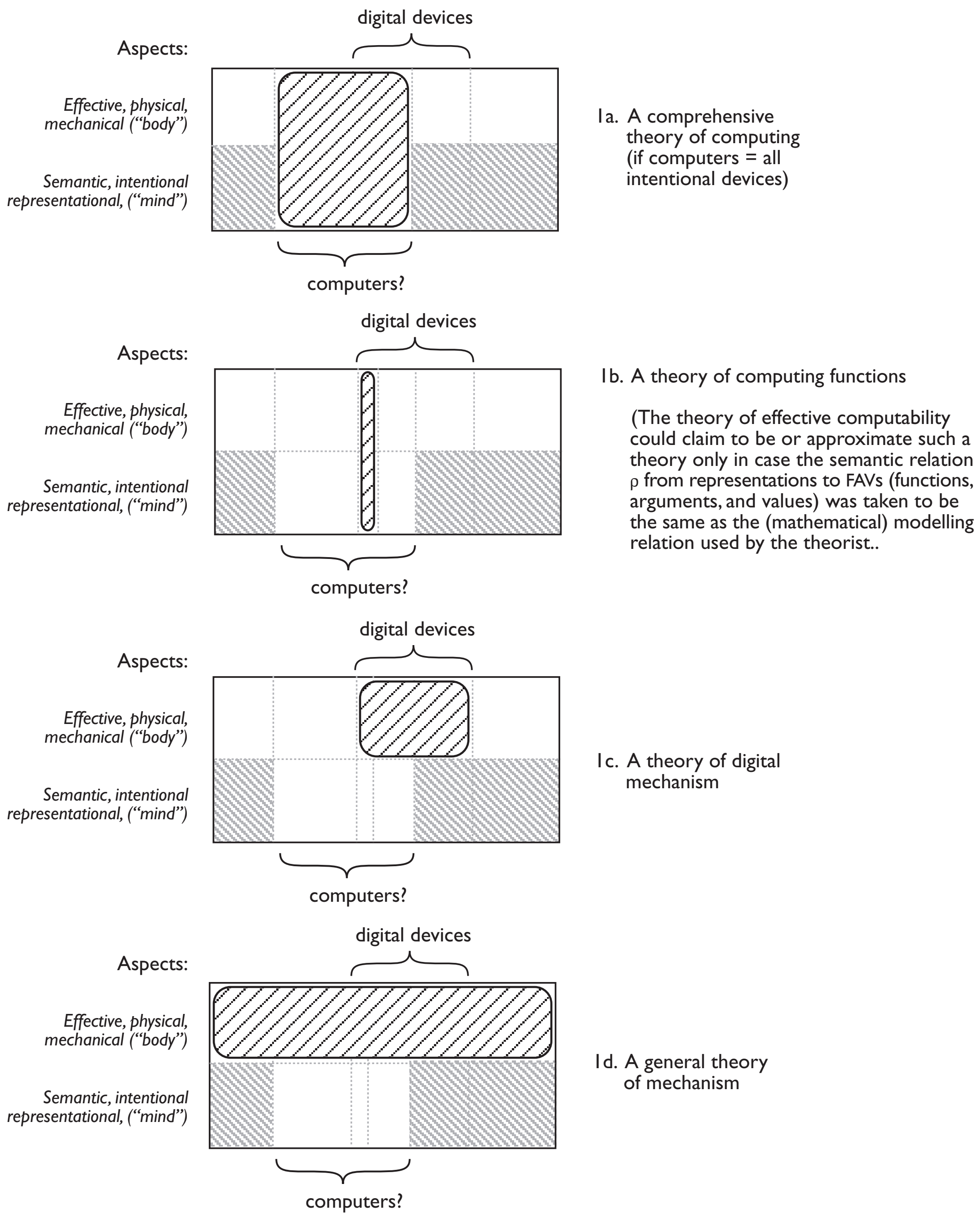
5. As we also pointed out:

> **P7** The 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 would have been, which claims it to be a theory of only a very particular species of computation: calculating the values of mathematical functions at argument positions (a species that will always be a specialized case).

D. This structure can all be understood in terms of the diagrams given in figure 1:[1]

[1]Note: this figure has the same contents as figure 1 of lecture notes 9b (March 8, p. 9·14), but the items have been re-ordered, for clarity.

1. What we are looking for—a comprehensive theory of computing, complete in **aspect** and **range**—is given in figure 1a.
2. The traditional understanding of the EC construal takes it to be something like 1b: a theory of computing, dealing with both mechanism and meaning, but restricted to the (narrower than general) case of the calculation of mathematical functions.


digital devices
Aspects:
Effective, physical, mechanical ("body")
Semantic, intentional representational, ("mind")
computers?

1a. A comprehensive theory of computing (if computers = all intentional devices)

1b. A theory of computing functions

(The theory of effective computability could claim to be or approximate such a theory only in case the semantic relation ρ from representations to FAVs (functions, arguments, and values) was taken to be the same as the (mathematical) modelling relation used by the theorist..

1c. A theory of digital mechanism

1d. A general theory of mechanism

Figure 1 — Coverage of various possible theories of computing

3. What we have argued that it *really* is given in figure 1c: a theory of the mechanistic aspect of all *digital* devices.
4. A *complete* theory of mechanism would be as indicated in figure 1d.

II. Three themes

A. What can we say we have learned, more specifically, from this critique? Three morals are especially important, which we can draw out.

B. One has to do with what we might call "**effectively-individuated types**"
   0. The issues we saw on the output tape of ★ are endemic to the entire situation
   2. The way we individuate *all* computational types—machines, inputs, outputs, etc.—build in assumptions of "reasonableness," not unlike the conditions explored on output marks
   3. Whenever we talk about machines—select reasonable encodings, define encoding functions (ρ), etc.—we implicitly rely on something that needs to be placed explicitly on the table.
   4. Will see some more examples of such "effectively-individuated" types today

C. Participatory theory
   1. Three dialectics
      a. Since the beginning, we have been tracking the **major dialectic**: between **meaning** *(*semantics, mind) and mechanism (effectiveness, body)
      b. We have also wrestled with what I called the **minor dialectic** (on the mechanism side): between *concrete* and *abstract* characterisations of effectiveness
      c. In lecture 3b, I noted two additional dialectics … static/dynamic, and one/many
      d. Recently, we have encountered a fifth dialectic: between **theory** and **subject matter**
      e. This fifth dialectic is fundamental to the current debate about the EC construal.
   2. Think about the discussion of the nature of the encoding function (ρ)
      a. One issue, on which we focused, had to do with which direction ρ runs: do marks represent numbers, or numbers represent marks?
      b. Another has to do with whether the mark-number relation ρ is part of the *subject matter*, or part of the *theoretical equipment*
      c. Mathematicians' claims that ρ is fixed, assumed, "not up for grabs" (i.e., not to be played with in the way that we played with it) make sense, I believe, under the reconstruction we came to: that ρ is *part of the theory*. If ρ were *not* part of the theory, then it would need to be understood as part of the subject matter, after all—and therefore dealt with generically (on the grounds that scientific theories should always disallow particulars).
      d. The fact that ρ is invariably *not* theorized—that it *is* taken to be fixed, assumed, not up for grabs—was taken as (partial) evidence for the claim that ρ was part of the theory, which in turn allowed the substantive claim: that, properly speaking, recursion theory, theory of effective computability, etc., are (mathematical) theories of the flow of effect.
   3. But it's too simple—this is the point—to take the theory/subject matter divide as so neat.
   4. FSM construal
      a. Think about what we concluded, during the first critique
      b. We started with the idea of a (syntactic) realm of symbols that was causally isolated from a (semantic) realm of reference

  . But then, after looking at computation-in-the-wild™, we claimed that such complete separation was untenable
  d. Instead, we argued for a much more *embedded, participatory* alternative
  e. Now it is time to lift that participatory moral to the level of the theory
5. Participatory theory
  a. Many people think of theories as sets of sentences, entailments of which are true
  b. Instead (as mentioned earlier in the semester) I subscribe to what I call an **integrity criterion**: a thesis that a theory of symbols or intentionality must be consonant, in use or form, with its own (subject matter) claims
  c. So a model of a theory as a detached, disconnected body of sentences, isolated from its subject matter, would be consonant with an FSM view
  d. But if we take a more participatory view of computing in general, what does that imply about our theory? In particular, does it suggest that our **theoretical perspective become more participatory as well**?
6. *Yes*, I will ultimately argue. (This is one reason why the result of the first critique is as strong as it is.)

D. Physical abstraction
1. One common response to this whole criticism is: "computers don't use marks"
2. Or another similar one: "for many purposes it is appropriate to consider the machine as dealing with *numbers*, instead of numerals"
3. But of course we didn't end up saying that this theory was (really) a theory of *marks*
4. Rather, the claim was that it was a *general theory of physical effect*
5. For the claim to be worth its salt, that is, "marks" must include *all physically effective states*
6. From the perspective of a theory of (the flow of) effect, what is evident, in all this, is that people are *striving to find a level of abstraction at which to specify effective regularities.*
7. Clearly, the subject matter is *more* abstract than the particular physical details
  a. We don't care (for many purposes) whether it is made of silicon, or gallium arsenide
  b. Also, most of the time, the theory of effect ignores a lot of issues like energy (power lines don't usually figure on the circuit diagrams)
0. Yet it can't be *so* abstract that connection with physical reality is lost (as in ★)
9. This "search" for an appropriate granularity at which to state conditions of effectiveness is (I claim) one of the main things that theoretical computer science has been aiming at.
10. That is: although we haven't found the EC construal to be answering (or even dealing with) our *major* dialectic, we *can* see it as struggling to find an answer to the *minor* dialectic, between the concrete and the abstract

III. The Rise of Programs

A. Theoretic specifying/modelling
1. As shown in figure 2, Turing machines are typically *modelled* or *specified* by the theoretician using a table, consisting of a set of entries, specifying state transitions.
  a. Each entry (canonically) contains the following information
    . A state σ

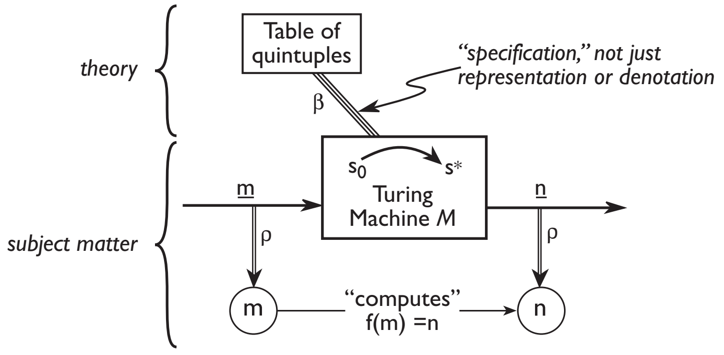


Figure 2 — Turing machines tables (standard interpretation)

ii. A mark τ
iii. A direction (right or left)
iv. A (new) mark τ'
v. A (new) state σ'

b. This means that if the "machine" (i.e., controller) is in state σ, and the mark under the read head is τ, then the machine moves right or left, as indicated, after writing the (new) mark τ' in the square under the head, and ends up in state σ'
c. Or something like that—the details don't matter here

2. What *does* matter is that the table itself, and the modelling relation β, are part of the *theory*
3. The relation β is semantical, one of *specification* or *modelling*
4. Together, the table and modelling relation must satisfy three constraints:
   a. It must (semantically) *denote* or *describe* or *specify* or *model* **M**
   b. What it must denote, etc. isn't everything about **M**, but **M**'s *effective behavior*
   c. It must (presumably) be *informationally complete*
5. I.e., if (as we're assuming) the effective and the semantic are the first and second fundamental aspects of intentionality, respectively, then the table must bear the *second* (semantic) relation to the *first* (effective) aspect of **M**.

B. Universal Machines
1. Consider what happens when one introduces the Universal Turing Machine **U**
2. A structure carrying the information about the machine **M**—i.e., the origin of what we will call the *program*—is encoded on the tape. See figure 3
3. What are the conditions on **p**, the mark that is the program?
4. Formally, the relations are described in terms of modelling or designation:
   a. The program **p** must *denote* (or encode) the number **p** that models (β) the machine **M**.
   b. Similarly, **m'** must *denote* (encode) the number **m'** that models (β) the input **m** that denotes the number **m**.
2. But is this right? *No!*
3. As before, if this was the only condition, then we could play with the modelling relations (β), so as

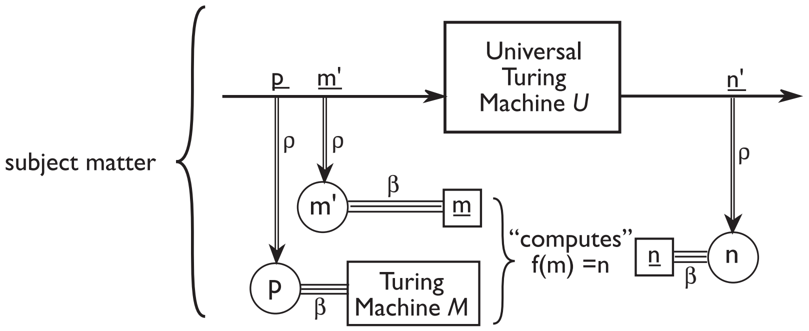


Figure 3 — Universal machines (standard interpretation)

to have informationally-complete models that didn't work

4. Rather, the constraint is that <u>**p**</u> has to *drive the machine to do the right thing*
5. I.e., the constraint is *causal* or *effective*
6. See figure 4

B. Programs

1. So we have our first condition on programs:

◆ A program <u>**p**</u> must honour a *double effectiveness condition*: it must *effectively specify* (i.e., in virtue of its effective properties, it must specify) the target machine's *effective behavior*.

2. That is, a program must effectively specify (to the interpreter or compiler or underlying machine) the effective behavior of the process to which the program is intended to give rise.
3. Which is to say, it should be able to **cause U** to behave like **M**.
4. This is the relation that we called the **program–process relation** (α) in the first critique.[2]

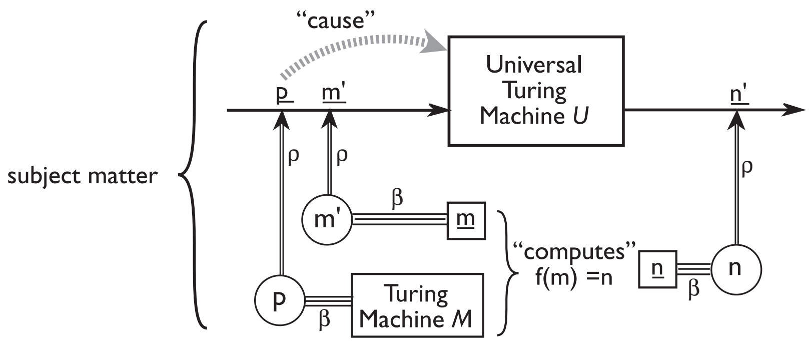


Figure 4 — Universal machines, reinterpreted

C. Status

1. Because of their *theoretic origins* (plus that fact that, *qua* observers, programmers, or theoreticians—that is, qua outsiders, we literally must understand them), programs are also taken to designate, model, or denote the effective structure of that same process
2. I.e., programs are supposed to bear *two* relations—one effective, the other semantic—to the *effective* aspect of the resulting process.

◆ Programs must be able to **cause** the behaviors that they (also) **specify** or **denote**

3. Hence the characterisation of programs as *prescriptions*
4. This is the origin of
   a. Why the word "interpreter" is used in computer science in the way that it is
   b. How operational and denotational semantics of programs came to exist
   c. Why Girard's linear logic has become popular
   d. … etc.

D. We will come to look at these things more carefully on Thursday.

—— *end of file* ——

[2]See also the "100 Billion Lines of C++" paper.