Copyright: © 2001 Brian Cantwell Smith Last edited Saturday, April 21, 2001

Programs (cont'd)

I. Preliminaries

A. Status
   1. We are running out of time. This is the fourth week (8th lecture) on the second construal: effective computability (EC). So according to the syllabus we should finish this critique today.
   2. We need to move on to Part IV of the course—on the fifth construal: digitality. But we won't get to it next Tuesday. In fact it is more likely that we won't start on digitality until one week late, on Tuesday April 3.

B. Plan
   1. Today I want to summarise what we did last time—lift our eyes a bit, to put it into context. The analysis on Tuesday was very detailed and complicated (and gets even more so, when one presses on details).
   2. Instead of going into detail, I want to:
      a. Review (very quickly) what we said about programs
      b. Talk (again, quickly) about programming language semantics
      c. Sketch what would be involved in developing a (mathematical) theory of the flow of (physical) effect.
   3. Also: summarize where we are on the overall project

II. The Rise of Programs

A. Review: we commented that our overall analysis is trying to make sense of three dialectics
   1. Since the beginning, we have been tracking the major or **primary dialectic**: between *meaning (*semantics, mind) and *mechanism* (effectiveness, body)
   2. We have also wrestled with what I called the minor or **secondary dialectic** (on the mechanism side): between *concrete* and *abstract* characterisations of effectiveness
   3. Recently, we've encountered another one, which we are calling a **fifth dialectic:** between *theory* and *subject matter*[1]

B. Programs
   1. Last time, we reviewed the origin of the notion of programming in Turing's paper—specifically, in the introduction of the universal machine
   2. Specifically saw (historically) how a program **p** has come to have a dual role:
      a. **Semantic:** it must *describe* or *denote* a specific machine, process, or behavior

[1] We also talked briefly about two others: a fourth, between what is *static* and what is *dynamic*; and a fourth, the ancient Greek issue between the *one* and the *many*.

   b. **Effective:** be a causal precursor of the (represented or desired) machine or behaviour. That is, it must be an "arrangement of marks"—or other causally efficacious organisational structure—able to cause or drive an underlying or universal machine to produce the desired (and denoted) effective behaviour
3. I.e., programs are supposed to bear *two* relations—one effective, the other semantic—to the *effective* aspect of the resulting process. That is:

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

4. Hence the characterisation of programs as *prescriptions*
   a. Aside: does that mean that both should be reconstructed under the 3rd RF construal?
   b. Perhaps. Leave that for another time.
5. We noted two salient differences between these roles

	Nature of relation of program to (desired or engendered) behaviour		Locus or realm of relationship
1)	semantic / representational	(⇒)	realm of the theory / theorist
2)	causal / effective	(→)	realm of subject matter

   a. Nature of the program-process relation borne to the (desired) behavior
      i. Representational role: a *semantic* relation (representation, description, etc.)
         α. I.e., "vertical" (double-tailed) in our standard picture
      ii. Effective role: a *causal or effective* relation
         α. I.e., "horizontal" (single-tailed) in our standard picture
   b. Locus or realm
      i. **Representational:** in the realm of the *theory*
         α. I.e., for us, *qua* theorists and/or observers
      ii. **Causal/effective:** realm of the computational *subject matter*
         α. I.e., within the scope of our theorizing
6. Also saw two salient *similarities* between roles (i.e., things true in both cases):
   a. Needed to be *"informationally complete"* in some appropriate sense
   b. The "target" of the program-process relation (whether semantic or effective) is the *effective behavior*—not the interpretation!—of the (desired or represented) machine or behavior.
7. Put it all together, as we said last time, in a slogan—using the word 'effective specification' to imply the combination of semantics, effectiveness, and information completeness:

> ◆ A program 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*.

III. Conceptual structure

- At this point we diverged from my prepared notes. They pick up next time (lecture 11a). What follows are some of the diagrams that were drawn on the board during the rest of this class.

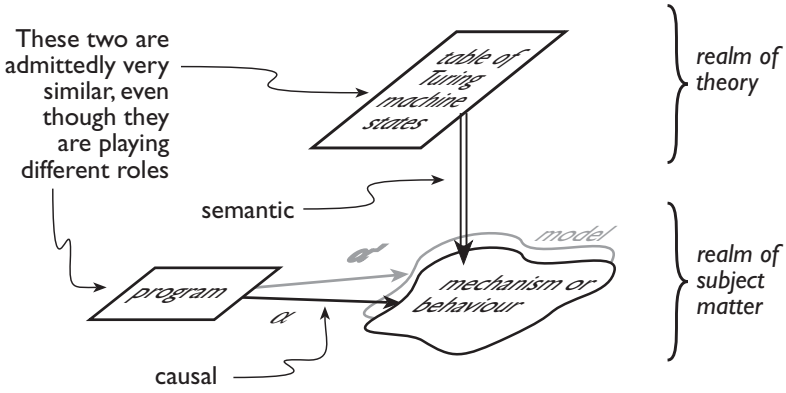

Figure 1 — Table and Programs

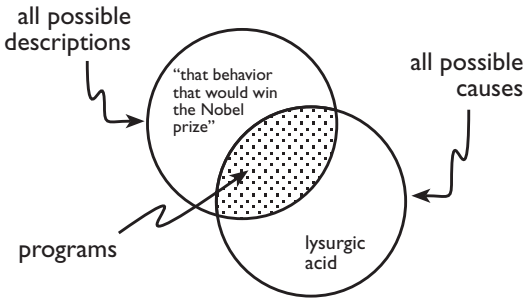

Figure 2 — Programs as descriptions + causal

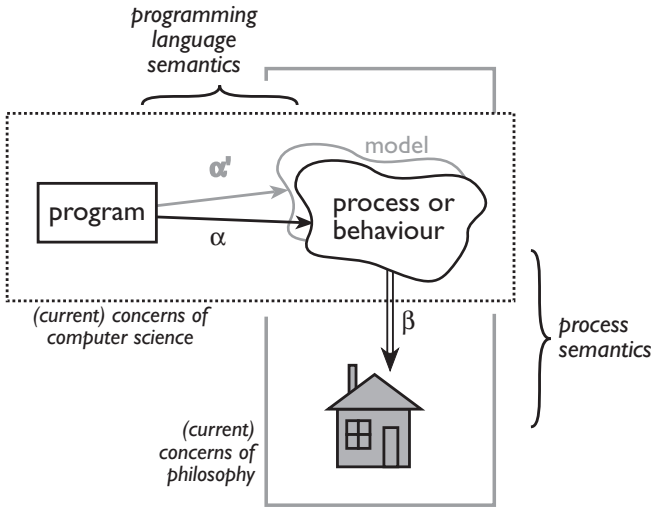

Figure 3 — Two "semantical" projects

—— end of file ——