 Last edited Thursday, April 19, 2001

Site, not Subject Matter

I. Preliminaries

A. Today is the last class of the semester, so we need to wrap up the whole discussion.

B. Review

1. Each construal says that computation was **special** in some particular way
2. That "specialness" was a necessary presupposition of the idea that there will ever be what we set out to find: a comprehensive theory of computing.
3. In each case, the form of the "specialness" claim took a very particular form:
   a. Computing was analysed in terms of a sensible substantive concept or distinction
   b. The construal assumed that the conceptual structure was *higher-order discrete*: pure, clean, absolute.
   c. The higher-order discreteness turned out to be constitutive of the construal's alleged **formality**, leading us to suggest that formality *means* "higher-order discrete"
4. In each case, our critique took the same form:
   a. The basic concept or distinction or issue that the construal dealt with was recognised to be genuinely important (essential to computation in the wild): the relation between mechanism and meaning, the relation between computation abstractly characterised and the nature or structure of the underlying physical realisation, the achievable degree of reliability, etc.
   b. What we challenged (in each case) was the formality claim: the claim that that relevant concept or distinction—in "computation in the wild"—was in fact formal (higher-order discrete) after all.

C. That leads to two conclusions[1]

> **C6 Computation is not formal** (at least not in the ways in which it has traditionally been claimed to be).
>
> **C8 Computation is not special**.

D. What we want to do today is to understand the consequences of each of these conclusions.

II. Formality

A. The formality result (**C6**) leads us is to something of a choice point:

---

[1]The numbers are from the last set of lecture notes: Week #15 (a), of Dec. 8, 1998.

1. **Conservative**
   a. We can hang onto to formality—i.e., not upset the *methodological* apple-cart, even if we are advocating strong changes in our theoretical understanding—by admitting that it is not formal in the ways in which it has been assumed to be formal, but may be formal in some other ways.
   b. This has the benefit of allowing us to retain mathematical analyses (in terms of the new conception, whatever it is).
   c. The challenge that this approach faces, however, is how to accommodate the very real lessons in the results we've arrived at.
2. **Radical**
   a. We can give up formality (higher-order discreteness) altogether
   b. This seems to do more justice to the empirical content of the results we've arrived at.
   c. But it is a very strong methodological (and metaphysical) result
   d. The challenge it faces is how, if one accepts it, one can remain open to any form of scientific or intellectual progress.

B. Approach
   1. My conclusion (based on much more analysis than we have had time to deal with in this short semester) is that the radical conclusion is right: computation is not formal *at all*.
   2. One argument for this is that it is *exactly formality* that has been understood (through all of our counterexamples) to fail to hold
   3. That is: all of our *substantive results* had to do with the breaking down of this or that allegedly discrete boundary: between the system and the world, between the "abstract" layer of computation and the underlying layer of physical implementation, etc.
   4. So it is not just that we found some other view of computing, that happened not to be higher-order discrete.
   5. Rather, it is *exactly the higher-order discreteness* that we have gradually and systematically dismantled.
   6. So to think that computation must be formal (higher-order discrete) *in some other way* seems vain: sentimental hope, not backed up by any empirical warrant that we have seen.

III. Site, not Subject Matter

A. Introduction
   1. The plausibility of this radical conclusion is made more palatable if we understand the consequences of the other conclusion—**C8**, that computation is not *special*.
   2. The overall situation—and some of its consequences—is depicted in figures 1 and 2.

B. Computation
   1. We started out assuming that computation was a distinct species or kind—represented by its containment with an Erlenmeyer flask in figure 1.
   2. *Within computing*, we assumed (as everyone does), there are a host of disciplinarily-specific concepts and categories (data structure, process, implementation, representation, etc.)
   3. Now we've argued computing isn't a distinct species or kind after all (we were misinformed)

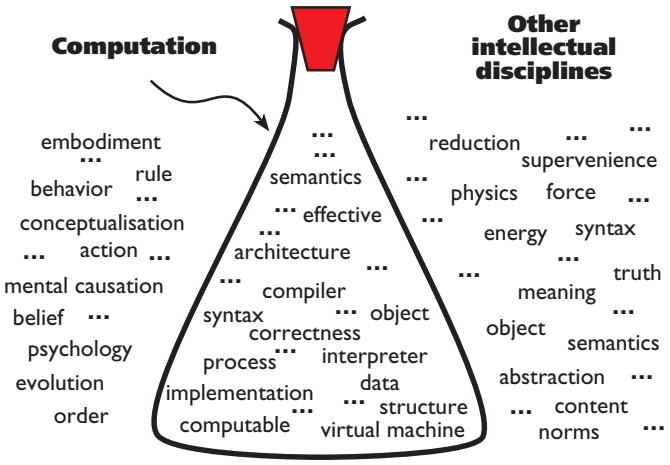

Figure 1 — Computation (as traditionally conceived)

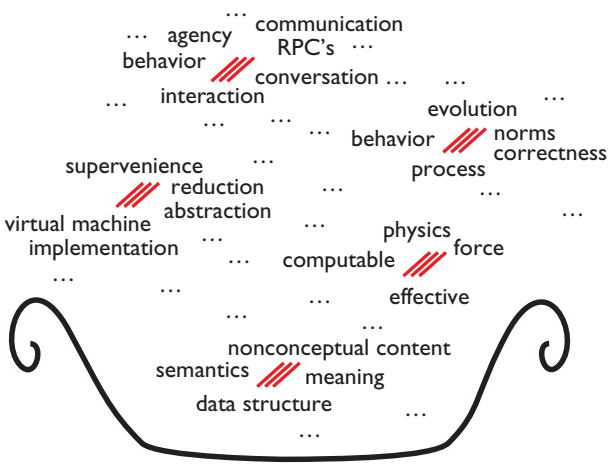

Figure 2 — Computation no longer confined

4. What that means is that all the allegedly-discipline-internal notions, such as those cited above, *aren't internal to a special discipline, after all*, because there is no such discipline for them to be part of.
5. So they float free, and have to take their place in general intellectual life, without the protection of a host or home territory (as depicted in figure 2)

C. Not a subject matter

1. What all this implies is the following strong thesis:

> **C9** Computation is not a subject matter

2. By 'subject matter' is meant a domain of inquiry that will repay intellectual investigation, sustain substantive and durable regularities, be the basis of an autonomous science or theoretical discipline … that sort of thing[2]
3. That is to say, the claim is that computers *per se* do not form such a basis.
4. So: in the long run there will be no such science as computer science.
5. There will never by any such science *because there is no subject matter there for there to be a theory of*.

D. Failure

1. Isn't this failure?
2. *No!* On the contrary, I believe that it is the *strongest possible positive result:*

> **C10** Far from being negative, the fact that computation is not a subject matter makes the 20th (and 21st?) century arrival of computing onto intellectual scene far more interesting and important than it would otherwise have been.

[2]Roughly, by subject matter I mean what philosophers might be tempted to call a "natural kind"

3. Why?
4. Because we are not saying that computing (in the wild) is intrinsically a-theoretical—and thus that there will be no theory of these machines, at all, when day is done.
5. Rather, claim is that such theory as there is—and there is as good chance of that as in any domain—will not be a theory of *computation or computing*.
6. It will be a full theory of intentionality: of embodied, significant, interpreted systems.

E. What are computers?

1. One question remains: what are computers?
2. There are two answers: substantive and methodological

> **C11** Computers (substantively) are *socially constructed intentional artifacts*.

3. So experience, skills, theories, and results of computer science are practical, synthetic, raw material for full theories of causation, semantics. I.e.:

> **C12** *The fact that computers are computational* has been a major block in the way of our understanding how important they are!

4. Also a methodological answer

> **C13** Methodologically, computers are a suite of examples of *middling complexity*, richer than the frictionless pucks and inclined planes of physics, but simpler than the full panorama of the human condition.

5. It is this that I have summarized in the slogan

> **C14** Computing is a **site**, not a **subject matter**.

F. Analysis

1. This last result (**C14**) explains something that we started with: how, with respect to all the foundational unification projects we talked about, computing is the "promiscuous bride" of intellectual life.
2. It makes sense because the subject matter (of computational investigation) is *all of intentional (epistemic, semiotic) science*
3. Computational experience is the ground for the first-ever theory of that realm

G. Semiotic alchemy

1. One way to understand computational *practice* is by analogy with the 17th century:
2. The sense of mechanism that we have been talking about is essentially the sense of the causal, physical, mechanical aspect of nature—the aspect that science has been studying for 300 years.
3. What we have been saying about computing, since the beginning, is that it is the study of

meaning along with mechanism. This introduction of the second of our two dialectics is a profound addition to the conception of nature.

4. On this conception, programmers can be understood as *epistemic or* **semiotic alchemists:** as knowing in their practice a wealth of details and intuitions that are the necessary forebears of true intellectual advance.
5. The rise of computing can be seen as ushering in a new age, comparable to the emergence of science in the 16th and 17th centuries, in which *meaning* and *mattering* will take their rightful place alongside *matter*, *material* and *mechanism* in our intellectual sense of nature.
6. I call this new age the **age of significance** (see figure 3).

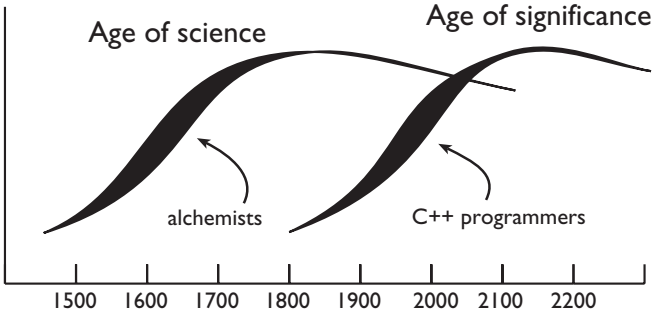

Figure 3 — The Age of Significance

IV. Formality (revisited)

A. Question: if computing is a site, not a subject matter, what do we have to say about the issue of formality?
B. My own answer is that computing is *profoundly* (metaphysically) not formal. That one way we can understand it is as a site which shows us the nature of a non-formal world
C. However, developing a theory of computing which can do justice to that non-formality is a non-trivial task.
D. That is the task that I attempt to answer in *On the Origin of Objects*
E. But because of this same (site, not subject matter) answer, developing that story takes us far beyond computing, into metaphysics full bore …

—— end of file ——