 Last edited Sunday, January 21, 2001

Mechanism
(The Problem of the "Body")

I. Review (Logic)

A. Intro
   1. Last time, we talked about logic
   2. We didn't go into technical details, focusing instead on its underlying conceptual structure.

B. Logic
   1. The conceptual structure is given in figure 1.
   2. It consists of
      a. A syntactic domain $\mathcal{S}$, consisting of sentences, identifiers, tokens, etc.
      b. A semantic domain $\mathcal{D}$, consisting of the entities that the syntactic items denote or signify or are about (objects, states of affairs, Truth or Falsity, etc.)
      c. A "derivability" relation ('⊢'), from sentences (or other syntactic items) onto sentences, defined in terms of the *syntactic* or *formal* properties of the syntactic items.
      d. An "interpretation function" ρ, mapping syntactic items onto what they are about (in $\mathcal{D}$);
      e. An "entailment relation" ('⊨') that, like derivability, maps from sentences (or other syntactic items) onto sentences, but is defined (vertically, in the diagram) in terms of the interpretation function ρ—and (horizontally) in terms of something like "truth-theoretic consequence" (or a subset relation) between (sets of) interpretations of the sentences.

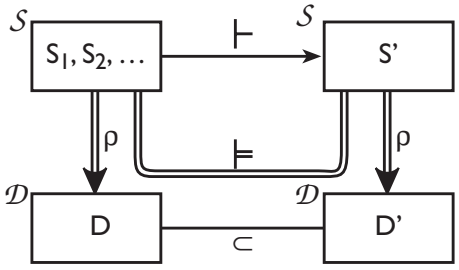

Figure 1 — Basic Logic

C. Constraints
   1. The most important conceptual point is the *restriction of the mechanical.*
      a. Note that the semantic relations ('⇒' and '⊨') must be perfectly well defined (in order for the logical system to be conceptually coherent)
      b. In addition, the syntactic relation ('→' or '⊢') must be *more restricted* than what is the case semantically ('⊨').
      c. If '→' *weren't* more restrictive than "⇒" then logic would be trivial—or vacuous: we could just *define* derivability ('⊢') to be identical to entailment ('⊨'), and all truths would be instantly derivable.

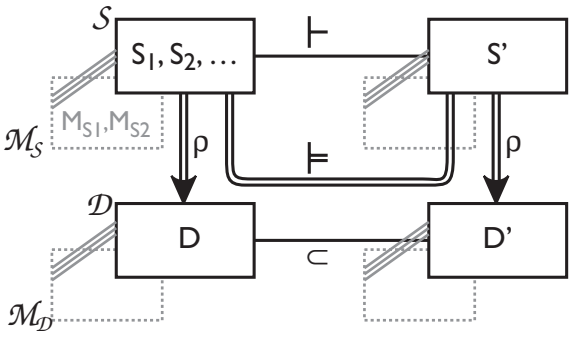

Figure 2 — Model-theoretic Logic

   d. In other words, the *conceptual preconditions conditions for the possibility of logic* (the possibility of it being a substantial topic) require that what is *(formally) derivable* be (intensionally) distinct from what is *semantically entailed*.

2. Norms
   a. Given those constraints, there are basically two norms:
      i. **Soundness:** that what can be derived is entailed
      ii. **Completeness:** that what is entailed be derivable.

B. Models

1. We will call the conception of logic we've been using so far (e.g., in figure 1) **direct**.
2. However logic is often studied **model-theoretically**, as indicated in figure 2 (previous page)
3. Instead of talking directly about sentences or formulae, and their types, one instead deals with **mathematical models** of those sentences.
4. Similarly, instead of talking directly about the denotations or references or interpretations of sentences, one talks in terms of *models* of those denotations.
5. Terminologically, there is a confusion:
   a. So far, I have used the term 'model' in a way in which it is used in English, and also (e.g.) in science—as, for example, in constructing a *model* of the solar system, or a *model* of gene replication, or a (computational) model of a tsunami.
   b. In logic, however, the term 'model' is used differently, in a very non-standard way.
   c. As indicated in figure 3, one talks (in logic) about a "model of a sentence" (or other syntactic entity) where really what is being talked about is a *model of the denotation or interpretation* of a sentence.
   d. It is absolutely vital to be clear on this distinction: the difference being one of semantics—the very subject we are trying to understand.

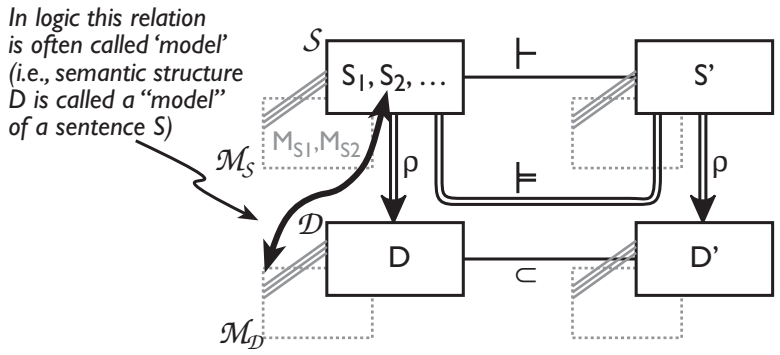

Figure 3 — The term 'model' in logic

C. Summary

1. This basic picture of the interplay of mechanical (syntactic, causal, effective—see below) phenomena and semantic (denotation, interpretation, etc.) phenomena is very important to understand.
2. Its development was also a stunning achievement, in 20th cent. philosophy & mathematics.
3. Keep in mind the fundamental structure and desideratum:

- It is the *semantic relations* ('⇒' and '⊨') that matter;
- It is the *causal relations* ('→' or '⊢') that do the work.
- The overarching desideratum is to have, or arrange for, the *causal relations and processes honour the semantic relations* (i.e., to have '⊢' mirror or match '⊨').

D. Diagrams
   1. For clarity and consistency, I have adopted a number of conventions in these diagrams, which we will continue to honour, throughout our discussions of computing.
   2. Mechanism
      a. The causal/mechanical/syntactic *operations* are in the "upper half" of the diagram, and run *horizontally*
      b. Causal relations (such as derivability) will normally be indicated with single arrows ('→')
   3. Meaning (Semantics)
      a. Mathematical or semantic states of affairs, in the *task domain*, are in the "lower half"
      b. Semantic relations (such as denotation) are indicated with double arrows ('⇒')
      c. By and large, semantic directedness ("denotation", etc.) will be indicated *vertically*.
   4. Note that these two conventions fit with the standard notations for derivability ('⊢') and entailment ('⊨').
   5. *Modelling* relations, in the ordinary English and scientific sense of the word (paradigmatically including modelling that is an epistemic phenomenon: part of the theory—i.e., as something like theoretical equipment) will be indicated with triple lines ('≡').

II. Mechanism

A. Introduction
   1. With this much of a discussion of meaning and semantics (last Tuesday), and this much of a picture of how logic works, and especially of how it embodies a solution to the fundamental and primary dialectic between meaning and mechanism, we need to turn to the other half of the primary dialectic: the issue of *mechanism*
   2. This is the "body" side of the "mind/body" problem for machines.
   3. It is also the dimension that I have referred to using the term "effectiveness"

B. Physicality
   1. By introducing effectiveness with respect to a notion of "body," I am treating it as if it were essentially *causal*.
   2. That is: I'm identifying effectiveness with *physical* or *material* properties (of a system)
   3. This makes sense for two reasons
      a. Intuitively sensible (ontologically)
         i. Some kind of materiality seems implicit in claim that computers are mechanisms
         ii. But that just says computers must be physically implemented to be *concretely real*
         iii. Deeper claim at stake: that very notion of effective computability (the core concept in terms of which the subject is traditionally defined) ultimately derives from physical properties of the underlying devices or systems on which the computations run.
         iv. I.e., that "computability" (whether something can be computed) is a *physical* issue
      b. Methodologically sensible
         i. Some such physicalist sentiment seems necessary in order to in order to **naturalise** computing: to show that it is continuous with the rest of natural science.
         ii. And if computation can't be naturalised, we would not have met the conditions for what we are looking for: a comprehensive theory.
   4. So it might seem like a natural move.

C. Problem
   1. But there is a problem
   2. Most current theories treat computing (or computability) and derivability as *abstract*
   3. This move is based on a thesis of **medium independence**
      a. Based on two underlying intuitions
         i. "Same" computation: carried out on variety of *different* (types) of physical device
         ii. Variety of *different* computations can run on "*same*"—i.e., single—physical device.
      b. First intuition more traditionally cited. But something like the second (which underlies notion of universality) plays an equally strong role in the tradition
      c. I.e., presumptive *two-way independence* (physical substrate vs. realized computation)
   4. Traditionally, therefore, formal definitions of computability make *no reference to concrete physical properties*
   5. Note: this strategy, too, looks overwhelmingly sensible
      a. Nothing *is* said—because seemingly nothing *can be* said—about:
         i. How much energy it takes to perform primitive operations in standard models ("add one," erase a mark on a tape, take "and" of two bits, clear a register, etc.)
         ii. Whether such operations happen mechanically, electrically, optically, or biologically,
         iii. Whether they are to be executed primitively by a microscopic gate, or sequentially by a computer the size of a star cluster
         iv. … and so on and so forth.
      b. Similarly, terms like "writing" and "erasing" are understood, if not quite entirely metaphorically (like the 'charm' of quarks), then at least in a very abstract sense.
      c. Similarly, notions of *space* and *time* are thought to be something like *abstract* or *metaphorical* (though which—and what either statement means—is very hard to say).
   6. Result: these considerations have driven theoretical computer science
      a. Away from physics and materials
      b. Towards abstract mathematics.

D. Status
   1. What is going on?
   2. What is the relation between *computational effectiveness* and *underlying physicality*?
   3. Should the "mechanical" aspect of computing be understood abstractly, concretely—or in some entirely new way (perhaps half-way in between)?
   4. What is the relation between a *computation* and a *computer*?
   5. Is a theory of computing going to be part of natural science, or part of mathematics?

E. Call these things **question of the body**

F. Similarly, we will take the implicit dialectic that this question deals with—between the *abstract* and the *concrete*—to the be **secondary dialectic** of computing. Dealing adequately with it—explaining all these various intuitions, resolving just what "level of abstraction" we want to talk to about computers and computation in terms of—this, too, will be an essential mandate on any adequate theory of computation.

III. Weak vs. strong abstraction

A. To start getting at what is going on, we will make a distinction between two kinds of abstraction
   1. Call something **weakly abstract** if it is (necessarily) concrete, but nevertheless constituted or defined at some "higher" level than that of elementary physical properties.
   2. Call something **strongly abstract** if it is not concrete at all—i.e., in the sense in which numbers, types, properties (and, according to some people, ideas) are thought to be abstract, i.e., not locatable in space-time, not *occurrent*

B. Examples
   1. *Weakly abstract*
      a. Hospitals (since they are constituted—individuated, identified—without concern for arrangements of proteins in the 2×4s in their walls), water molecules, whirlpools, traffic jams, détente, consumer confidence, the particularly-configured strike zone through which the ball passed that Joe McGuire hit as his 100th home run.
      b. Note that not all of these things are *echt physical objects*.
      c. Some lack physical heft (détente, consumer confidence, strike zones, whirlpools?)
      d. Nevertheless, they are all still perfectly concrete, in this limiting sense: if you removed (or blew up) the physical universe, they would go away. (Post-explosion, you would describe them in the past tense.)
      e. In that sense they differ from numbers.
   2. *Strongly abstract*
      a. Numbers, sets
      b. Properties, relations
      c. "Concepts" and "Ideas" (in the sense in which you and I can share the very same (numerically-identical) concept or idea.
      d. Types—including types of physical object. Thus the type 'tiger' and the type 'city' are strongly abstract, no matter how concrete individual tigers and cities are.
      e. Words (as opposed to token inscriptions, or concrete utterances, of words), since we normally talk about words as types.

C. Discussion
   1. Weak abstraction is *extremely weak*. That's the point. Just about everything—including, arguably, every concrete physical object—is at least weakly abstract.[1]
   2. Only strong abstraction is opposed to concreteness.

D. Computation
   1. Our question: is the kind of abstraction relevant to computational effectiveness strong, weak—or some third type?
   2. In particular, when a "computation" is (in traditional theory) identified as an *abstract object*, which of the following is intended:
      a. That it is a *strongly abstract particular* (individual)?
      b. That it is a *weakly abstract type*—in the sense of being a type (hence itself strongly ab-

[1] At least every extended or reidentifiable concrete object—i.e., every object other than the putative space-time points that physicists use to describe fields. I myself do not believe that space-time points are objects, however.

stract) of *weakly abstract* (but nevertheless fully concrete) *things*—namely, concrete computations, or concrete computational *processes*?[2]

c. That it is a *strongly abstract type*—in the sense of being a type (and hence itself strongly abstract) of individuals, each of which is also strongly abstract?

d. Or something else entirely

3. *Nothing in current theories answers this* question.
4. This is a stunning ontological omission.

IV. Potency predicates

A. Setup

1. Over the years, people have used a spate of different predicates to get at computation's first (mechanical, effective, "do-something") aspect.
2. Thirteen of the most important are listed in figure 4.
3. Call them **potency predicates**[3]

B. Potency predicates

1. Physical

a. Three—*physical*, *causal*, and *material*—clearly and unambiguously have to do with a system's physical embodiment

b. Two—*local* and *internal*—depend on the existence (establishment) of a boundary between the system and its environment, and boundaries presumably have to do with the *space* occupied by the system, a physical (at least geometrical) notion.[4]

c. One—"like shape"— in reference to Fodor's claim that computational operations "apply in terms of the, *as it were, shapes* of the" system's internal structures,[5] and that "syntax [see below] essentially reduces to shape." Whatever else shape is, it presumably has something to do with causally efficacy.

Potency predicates

A. Physical

1. Physical
2. Causal
3. Material
4. "Shape"
5. Internal
6. Local

B. Functional

1. Functional
2. Mechanical
3. ✔ Effective
4. (Computational$_1$)

C. Linguistic

1. Formal
2. Syntactic
3. Grammatical

Figure 4—Potency Predicates

---

[2]Note: throughout, I will use "weakly abstract type" and "strongly abstract type" to refer, respectively, to things that are strongly abstract: types of weakly abstract thing, and types of strongly abstract thing.

[3]As usual, I take *predicates* to be representational items—typically, elements of language—and *properties* to be ontological features of the world that predicates denote. I am calling this a list of potency *predicates*, rather than potency *properties*, because the entries reflect distinct ways of speaking and describing the world. It is far from clear what (classes of) properties each entry denotes—whether they are the same, overlapping, distinct, or even refer at all. Figuring out the semantics of the entries in the list is one way of characterising the whole project.

[4]Some people will prefer to include 'local' and 'internal' in the second, functional group, on the grounds that strongly abstract systems can still have "insides" and "outsides" That's fine; nothing hangs on this taxonomy. My only concern is with the warrant for the boundary: who draws it? what physical or metaphysical or conceptual distinction does it represent? Is it justified in the world, or an epistemic projection of the theorist?

[5]Ref Fodor. Note that he assumes that the internal elements are symbols; for now, not a lot depends on that.

2. Functional[6]

a. Three—*functional*, *mechanical*, and *effective*—that lean towards notions of role or function, away from detailed facts of material embodiment.

b. Also 'computational$_I$'—i.e., the use of 'computational' in a single-aspect sense, like 'syntactic', to refer to computation's effective or mechanical aspect, *as distinct from its semantic side* (common in philosophy of mind—but bracketed to indicate my claim that this usage is essentially mistaken).

3. Linguistic

a. Three—*formal*, *syntactic*, and *grammatical*—that seem applicable only to things like words, sentence tokens, formulae, and expressions in a formal language.

b. These terms, though, are perhaps the most common potency predicates used to describe computation, so they will be of special interest to us here.

C. Metaphysical properties

1. Introduction

a. Figure 5 lists another (fourth) set of related types of property: **metaphysical**, having to do with what's necessary to something's being the thing, or being the kind of thing, that it is.

b. Three: *essential*, *intrinsic*, *constitutive*

Metaphysical
1. Constitutive
2. Essential
3. Intrinsic

Figure 5

2. Discussion

a. Relation

i. No means obvious that intrinsic or constitutive properties should have anything specially to do with effectiveness

ii. Nevertheless, *many views about what computing is*, and about *how it should be studied*, involve ties between and among these sets, including these metaphysical ones.

iii. E.g.: overarching presuppositions of physicalism mandate that the properties in virtue of which computers *do things* must be intrinsic properties, on pain of metaphysical inconsistency.

b. Example: Searle's second argument against the computational view of mind:[7]

i. Computers work in virtue of syntactic properties

ii. Syntactic properties are not intrinsic properties

iii. Consciousness is an intrinsic property

iv. Therefore consciousness cannot be computational

c. Or (an implicit parallel set of assumptions that he also seems to believe):

i. Computers work in virtue of syntactic, hence non-intrinsic, properties

ii. Natural kinds are constituted by intrinsic properties

---

[6]Note (for philosophers): *functionalism* was (supposedly) a doctrine, about the mind, according to which mental states are individuated "functionally," on supposed analogue to how computational machines states are individuated. What is not clear, however, from the philosophical discussion of functionalism, is what kind of individuation functional individuation is. One of the conclusions of Part III of the course will be that functionalist individuation is usually radically more physical (concrete) than is typically recognised.

[7]This is the argument that he describes as showing that *syntax does not inhere in the physics*. The more famous first argument, involving the Chinese Room, is an argument to the effect that *semantics does not inhere in the syntax*.

iii. Science, which studies natural kinds, is only interested in intrinsic properties
iv. "Being a computer" cannot be a natural kind, since it is not defined in terms of an intrinsic property
v. Hence *being computational* cannot be scientifically explanatory

3. These sorts of chained assumptions (implicit arguments) we at least to expose—or rout.

D. Strategy

1. We will select '**effective**' as the potency predicate of choice.
2. That is, we'll *stipulate* that computers work—do things, accomplish what they accomplish—in virtue of their *effective* properties.
3. I.e., from here forward, we will refer to computing's first aspect as its *effective* aspect: its activity- or result-oriented dimension, the aspect of its character having to do with its being a *mechanism*.
4. This is a modal claim: not just assuming (i) that computation is, in fact, effective, but (ii) that computing is what it is (in part) in virtue of that effectiveness.

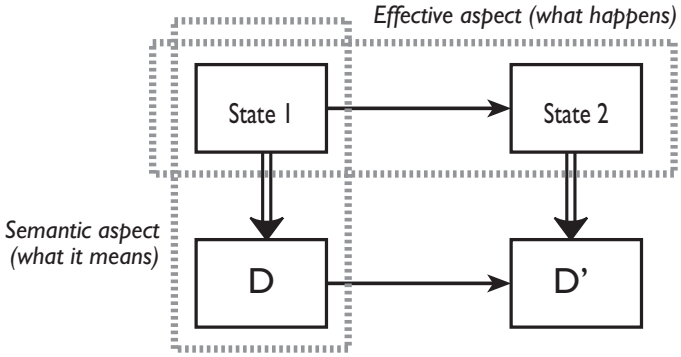

Figure 6 — Two aspects of computing

5. In sum, situation is as in figure 6
   a. I.e., will use 'effective' where some (e.g., philosophers of mind) would have used a univalent sense of 'computational' (i.e., what I am calling "[computational$_I$]")
   b. They would have said "A theory of mind will consist of two components: a computational account of how it works, and a semantic or intentional account of how it has (broad semantic) content."
   c. I will instead say "A computational theory of mind should be expected to consist of two components: an *effective* account of how it works, and an *intentional* or *semantical* account of how it is has (broad semantic) content."

E. **Question of the body**

1. The question of scientific interest is what effectiveness is.
2. Some different ways of asking it (at root really just one question):
   a. What is the metaphysical origin of the classic "computability constraints" (which I will eventually rename "effectiveness constraints")?
   b. Does computational effectiveness derive from physics?
      i. If so, then how—and to what degree?
      ii. If not, then how are the two related?
   c. What identity and individuation conditions underlie the notion (or notions) of the "same computation"? What is the right (strong or weak) notion of abstraction in terms of which to formulate it?
   d. How should the principle of (partial or full) "medium independence" be explained—the idea that the "same computation" can be implemented on different kinds of hardware,

and that different computations can run on the same piece of hardware?

e. What is the relation between a *computation* and a *computer*?

f. Will a theory of computing be part of natural science, or part of mathematics?

i. If the former (part of science), how is that to be rationalised with results in logic and mathematics (e.g., about recursive functions, abstract computability, etc.)?

ii. If the latter (part of mathematics), what does that imply for sciences that rely on notions of computing in their own theorizing—such as the computational theory of mind? Do current mathematised theories of computability implicitly support a kind of property dualism?

3. Equivalently

a. Another way to task the question:

i. Did God, in creating the world, made *two* kinds of efficacy?[8]

α A concrete or physical one, having to do with energy, materials, and literal pushing and shoving—what is normally called *causal* efficacy? and

β An abstract one, having to do with computability, which we are calling *computational* effectiveness?

ii. Or did God provide efficacy in just one form—suggesting that the latter may be merely a reflection of the former, perhaps in abstract guise?

b. This study will ultimately settle on the latter answer: there is just one.

i. Ontologically, this is the preferable result: it paints a picture of a simpler world than would have been the case, had the answer been two.

ii. But conceptually and explanatorily not so simple.

iii. Will require showing (among other things) that all of computer science's fundamental theorems—the halting theorem, Gödel's incompleteness results, results about computational complexity bounds, etc.—can be reformulated in terms of, and derived from, a (perhaps reconstituted) physics.

V. Plan

A. Summary

1. OK, this is enough introduction to the issues
2. Starting Thursday, we'll come back to computation, and start looking at the first construal of computing: that what it is to be a computer is to be a **formal symbol manipulator**.

—— *end of file* ——

[8] I'm not suggesting that I believe in God; this is just a rhetorical device.