CSci • B607 **Week #7 (a)** **Feb 20, 2001**

Copyright: © 2001 Brian Cantwell Smith **Last edited Sunday, February 18, 2001**

Part III — Effective Computability

◆ *Note: There will be no class this Thursday, February 22, 2001. See I.F.2, below.*

I. Preliminaries

A. Introduction
   1. Last time we finished our analysis of the first construal: formal symbol manipulation (FSM)
   2. FSM came first in several senses
      a. Not only was it the first construal; it was also framed in terms of the first *dialectic*: between meaning and mechanism.
      b. Historically, too, it deserved pride of place, because it represents (essentially *is*) the intellectual structure of the logical tradition, whose intellectual apparatus still forms the background skeleton in terms of which present-day theories of computation are framed.
   3. Today we start Part III of the course, looking at the second construal: **Effective Computability** (EC). It includes within its scope such familiar notions as:
      a. Turing machines, computability theory, complexity theory, etc.
      b. Programming language semantics (denotational and operational), linear logic
      c. The so-called (official) "theory of computation"
   4. As a result, will be
      a. *More* familiar to computer scientists (than the first construal was)
      b. *Less* familiar to cognitivists (AI, cognitive science, philosophy of mind)

B. Status
   1. Just as it made sense to consider FSM first, for several reasons; EC comes *second* in more than one way. Not only was it ranked second in the list of construals; analysing it will force us to focus, in a concentrated way, on the *secondary dialectic*: between the abstract and the concrete.
   2. Historically, it also makes sense to consider EC second. In dealing with computability, Turing machines, recursion theory, etc.—it represents the theoretical approach to computing that most immediately developed (out of the logical and metamathematical traditions) at the beginning of computer science.
   3. Maturity
      a. On the face of it (because of the extensive mathematical results), the second construal also looks more like a mature science
      b. Note, too, that this construal doesn't mention "computing" or "computation" explicitly[1]

[1] Just as physics doesn't mention "physical"?

c. Rather, proposes a cluster of notions—effectiveness, "computable," universal, complexity orders, etc.—in terms of which to analyze practice.

C. Discussion

1. There is a way in which one can view the second construal as continuous with the first.
2. As I pointed out on Thursday, we didn't have time to get back, in anything like adequate detail, to the **conceptual** sub-reading of (the negative reading of) formality: the metatheoretic claim that computation is "formal" just in case one can give an *account* of how computing works independent of—i.e., "without reference to"—a semantical account (or, which may or may not be the same thing, an account of its semantics).
   a. As we noted, some people—such as Fodor, perhaps even Haugeland—seem to think this is true of computing, because semantics is an "addition to" the effective workings.[2]
   b. We did note that even if this were true, that wouldn't constitute a complete account of computing (on the first construal), since (almost by hypothesis) it wouldn't be an account of its semantics. So it would be something like a "half of a theory of computing." I.e., it would still fail what we called the **aspect** criterion[3]—that a comprehensive theory be an account of all criterial (essential) aspects of computing, not just some.[4]
3. In addition, even if the theoretical structure turned out this way—so that the account of how a system "works" (mechanically) is separable from the account of how it means—the result would remain almost completely programmatic. In order to have anything count as an actual theory of computation, one would still need to *give such an account* of the workings—i.e., give an account that did not "make reference to" semantics, but that nevertheless *did explain how computers work*.
4. Haugeland's proposal (in "Semantic Engines" and in *AI: The Very Idea*) that computers are "automatic digital systems"[5] gestures towards such an account. But of course it is very elementary; if it were the right approach, it would need to be fleshed out in considerable detail.
5. One way to understand the account of computing we will look at now—i.e., the second construal, the story about effective computability, framed with reference to Turing machines and the like—is *as such a (semantic-free) account*.

D. Plan

1. The investigation will, as usual, have three parts:
   a. **Conceptual** phase: to understand what the second construal is saying
   b. **Empirical** phase: to see whether it is true (of computing-in-the-wild)

---

[2]Cf. Haugeland's claim that computation = an automatic (formal) digital system *with an interpretation*.

[3]See the notes for lecture 4a, on January 30, 2001.

[4]Given the positive (participatory) morals we learned, about internal exemplification of semantically designated properties (e.g., lengths of lists), it would appear that there are internally-exemplified semantical relations (or at least we have seen no reason to suppose that there could not be). It follows, therefore, that an account of how the system works that does not traffic in semantics is liable to fail the **level** criterion as well. When we get to criticising the ontological status of computational systems, we will use the functional individuation of internal states to argue that semantic criteria are employed in identifying data structures, etc.—so this "level" threat turns out to be true.

[5]Where "digital" means finitely playable, perfectly repeatable, etc.

   c. **Explanatory** phase: to figure out how its claims fit into the wider intellectual landscape.
2. As usual, whereas the "foil" or "official strategy" of the investigation will be to determine the adequacy of the proposed account, our real goal will be to use this as a device to find out *what computing is actually like*—by ferreting out our intuitions, seeing what aspects the official theory deals with and which aspects it does not deal with, etc.
3. Ultimately—i.e., by way of conclusion—I will argue that the theory of effective computability, as I will call this construal:
   a. Is *not* a theory of computing at all (it is mislabeled)
   b. *Is* a theory of "general effectiveness" (i.e., a theory of **causality**).
   c. I.e., that a theory of computing isn't really a theory of *mathematical functions defined over numbers*, represented by the marks on the tape.
   d. Rather, it is the opposite: a theory of *marks on tapes*, represented by mathematical functions over numbers.
4. I will argue this conclusion with a dramatic example: machine that solves the halting problem
   a. It doesn't *really* solve the halting problem, of course
   b. Rather, the claim will be that it satisfies the official criteria that have been laid out as what it would be to solve the halting problem
   c. So what it will be used, here, to show is that those formulated criteria aren't right (or anyway aren't complete)
   d. It is in fixing them that we will be led to the "nugget" we are going to extract from the failure of this construal.
5. Because of that strategy (and because we will end up rejecting a great deal of how the subject is normally understood), we will have to set things up exceedingly carefully.
6. So we will take more than the usual care in approaching the subject.[6]

E. Today, I want to make a dozen preliminary remarks, of three types:
   1. Four methodological
   2. Six substantive (easy)
   3. Two substantive (harder)

F. Administrivia
   1. Problem sets:
      a. The first problem set has now been graded. Comments and grades are posted on Annotate.
      b. The fourth problem set has been posted. It is due one week today, on February 27.
   2. No class on Thursday! There is a conference, held here on campus, starting this Thursday, that I will be attending and participating in. As a result, there will *no class this Thursday*, February 22. Our next class will be one week from today, on Tuesday Feb 27.
   3. Readings: some of AOS Volume III is now available for downloading on the class web site:
      a. AOS Volume III Chapter 1 ("Turing Machines")

---

[6]Note: I am presuming that everyone is familiar with Turing machines, effective computability proofs, etc. This setup is meant as a review, not as an introduction.

II. Four Methodological Remarks

A. Formal and informal models
   1. Turing machines are normally introduced in two steps
      a. **Informal:** tape, machine, controller, a few examples
      b. **Formal:** mathematical theory (with sets of quintuples, formal results, etc.)
   2. It is important to realize that there is no guarantee that these two models will align
      a. That is: it's contingent whether the *formal* theory adequately captures the *informal model*
      b. This raises the question of which of the two "is right." There are two possibilities:
         i. The *informal* one—because that is what grounds intuition and captures imagination
         ii. The *formal* one—because that (after all) is what is "formally" or rigorously defined, in terms of which the enormous subsequent body of mathematical theory is framed.
      c. I lean towards the former, but I am not committed to one over the other
      d. What will matter, here, is that we keep the two *rigorously distinct* (since, as I will show, there are enormous conceptual distinctions between them).
   3. Example: the representational status of tape
      a. We often understand some simple algorithms in terms of a mark being left *to mark the square at which the controller started* (to which it may then return at the end)
      b. That square isn't theorised as such—i.e., isn't theorised as "*a representation of where the controller started*" in any formal model. That is: no mathematical theory that defines an interpretation function over the marks (or configurations of marks) on the tape ever maps that mark *onto the square where it is written*.
      c. Does that mean it *isn't* a representation? No—not at all! It may *be* such a representation (for the controller or machine, even—not just for us). All that we can conclude, from the fact that it is not treated as a representation by the official theory, is that, if it is a representation—even if it is *essentially* a representation—that fact isn't captured by the theory we are used to. (Perhaps a better or fuller theory *would* or *should* treat it as representational)
      d. Indeed, this kind of (tape-to-tape) representation may be critical to the device's computing what it does—or critical to our taking the machine to have computed what we take it to have computed.
      e. Yet it is left out by official theory
   4. So that's the first methodological remark: we must not confuse or conflate:
      a. *What current theories theorise* and
      b. The full suite of phenomena that are going on, in the informal model—and that may be critical (i) to the machines being a computer, and (ii) to our (human) understanding of it as computing the function that we think that it does.

B. Equivalence
   1. As mentioned in passing in earlier lectures, a very famous notion of "equivalence" underwrites a great deal of the official theory of computation, including the theory we will be looking at here, defined (in terms of the formal theory) can "compute the same function(s)"
   2. For example, proofs may claim that a machine α has been shown *equivalent to some other machine* β when it has been demonstrated that α can *compute the same function as* β.

3. This equivalence then used in other forums (e.g., in the Church-Turing thesis)
4. What's crucial to understand is that this is a *theory-relative equivalence metric*
5. On a different view of computing, two architectures that are equivalent under the Turing-theoretic view may not be equivalent, under the new one!
6. For example: a view that distinguishes *being* and *implementing* a virtual architecture. Or (perhaps) a view that is not *semantically blind*
7. That is: the notion of equivalence used in these analyses is essentially blind to the "crosses implementation boundary" issues that were explored in the second problem set.
8. For now (this is the second methodological remark): we need to be careful in using any notion of equivalence, to recognize that this is theory-laden talk—laden, in fact (this is why this is so important) by a theory that I will soon argue is not a theory of computing after all.
9. Such differences are a source of potential distraction
    a. Turing machines *as imagined* (informally) may possess a variety of characteristics that are genuinely true of computers
    b. If we *assume* that the (formal) theory of Turing machines does justice to these (informal) intuitions, we may (uncritically) assume that the theory of Turing machines is right or intuitively complete
    c. But no such conclusion follows!
10. In sum: the (formal) theory of Turing machines may fail in either (or both) of the two ways (i.e., first, as well as second, is a possibility):
    a. As a theory of (informal) Turing machines
    b. As a theory of computing more generally.
11. This is complicated by spate of facts:
    a. Few concrete Turing machines actually exist (they're "often imagined, but seldom seen")
    b. It is easy to "implement" or "simulate" a Turing machine on an ordinary computer
    c. Notion of a universal machine blurs the boundary between *being* and *simulating* a machine $\Leftarrow$ this is absolutely critical
    d. Because of the famous equivalence proofs, many people call all computers Turing machines, because they are "equivalent in power"
12. In sum
    a. For purposes of a comprehensive and conceptually adequate account of computing, it is essential to distinguish
        i. One (virtual) machine, from
        ii. Another that is implemented on top of it, or another on which it is implemented
    b. Even if familiar notions of equivalence blur the distinction between them.
    c. Keeping such machines apart need not blind us to the fact that one way to get one machine (α) to do something is to implement another machine (β) on top of it.
    d. That is: maintaining a clear distinction between such virtual machines need not obscure the tremendous power of (the notion of) implementing one machine in another.

C. Indirect classification
1. As said before (in Part I), the model-theoretic approach to logic make a distinction between:
    a. **Direct analysis:** in terms of actual (concrete?) things being studied: sentences, semantic

subject domains, (intended) interpretation, etc. (figure 1)

b. **Indirect analysis:** analysis framed in terms of (typically mathematical) *models* of at least the subject matter, with correspondingly refigured interpretation relations (figure 2)

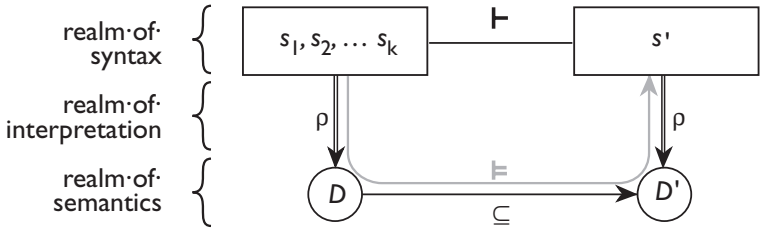

Figure 1—Logic, studied directly

2. In this critique we will be focusing on the notion of effectiveness
   a. Cf. potency: shape, cause, syntax, etc.
3. In doing this, we need to distinguish between
   a. Properties (including efficacy) of the *primary subject matter*
   b. Properties (including efficacy) of the *mathematical models* of that subject matter
4. Example: 17,000 mph
   a. Velocity may be an effective property
   b. The number 17,000 is not effective, even if we use it to classify a property that is.

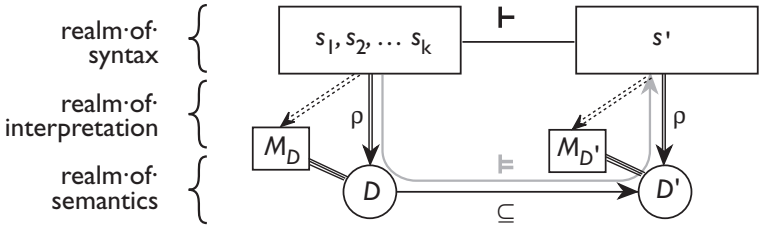

Figure 2 — Logic, studied indirectly (via models)

5. Why raise these things?
6. Because I will claim that:
   a. Missteps about the nature of the classificatory situation …
   b. Specifically, about which phenomena are classifiers, and which are classified …
   c. Have blocked us from understanding the Turing-machine model …
   d. Thereby hindering us from understanding its relevance to genuine computation

D. Formality
   1. The fourth methodological remark has to do with **formality**
   2. As I mentioned in Part I, issues of formality are very important to me, and in some ways (behind the scenes) have structured this entire investigation. That is: asking whether computing was *formal* was one of the first questions I had, with respect to the whole subject matter. As it turned out, though, there isn't any one reading of formality that makes that a possible theoretical entrée into the subject matter.
   3. Review
      a. In analysing the first construal, we encountered two basic meaning of formality:
         i. A positive one, having to do with syntax, effectiveness, and other potency predicates
         ii. A negative one, meaning "independent of semantics"
      b. In our analysis of the second construal, issues of formality will once again come to the fore.
      c. Since we will be concentrating on efficacy, in the notion *effective computation* (or effec-

tive computability), you might reasonably expect the first of these to be the relevant one—i.e., to be on center stage

d. You especially might think this given that we redirected the positive reading of formality to our analysis of this second construal.[7]

4. In fact, however, although we will study efficacy here, and use the results of this analysis as constituting our understanding of the positive reading of formality mentioned above, when the term 'formal' is used, in this context of these theories, that is not what the term is used to refer to.
5. Rather—as suggested in the distinction between "formal" and "informal" models of Turing machines, mentioned above (§II.A.1, above)—"**formal**," in these contexts, is usually taken to mean **mathematical**.
6. So this is our *third major understanding of the term formal*.
7. Why the term "formal" has such apparently disparate readings—how it happened, historically, whether there is anything in common among them, whether any underlying unifying principle unites them—is a question that will be receive its answer in a major ultimate result of the entire investigation, which we will get to at the very end of the course, at the conclusion of Part IV (on digitality).

E. Summary: So those are the four methodological remarks:
1. Keep formal (mathematical) and informal (intuitive, concrete) models of Turing machines apart
2. Distinguish one virtual machine from another that is implemented on top of it, or in terms of which it is implemented.
3. Distinguish (i) the actual subject matter of a theory from (ii) a mathematical model of that subject matter. (The former can be got at with what we are calling a *direct* analysis; the latter, with an *indirect* or *model-theoretic* analysis or account.)
4. Keep an eye on the reading of 'formal' as "mathematical," while at the same time recognising that the notion of efficacy or effectiveness to be studied in analysis this second construal is meant to do double duty, not only to explain the official mathematical theory of computability, but also to reconstruct the "positive" reading of formality we encountered in analysing the first construal (formal symbol manipulation).

F. Given those methodological preliminaries, turn next to six relatively simple substantive remarks about Turing machines.

III. Turing machines I: Six (simple) introductory remarks

A. Machines, controllers, and systems

1. There is a small semantic ambiguity about what the term "Turing machine" refers to (fig. 3)
2. Has in part to do with how Turing machines are *individuated*
3. Is the Turing machine the *controller*? or is it the *controller plus the tape*?
   a. If "plus tape," is it with specific configuration of marks on the tape?
   b. Or with a tape "in general"?
4. In this class: I will take the **Turing machine** to be the *controller*

---

[7] E.g., see figure 1 on page 2 of the notes for lecture 4a (of January 30, 2001).

   a. Ask, e.g., whether Turing machine $\mathfrak{M}$ will or will not, given a certain input
   b. Use the term **system** to refer to the amalgam of *controller* (Turing machine) plus *tape*

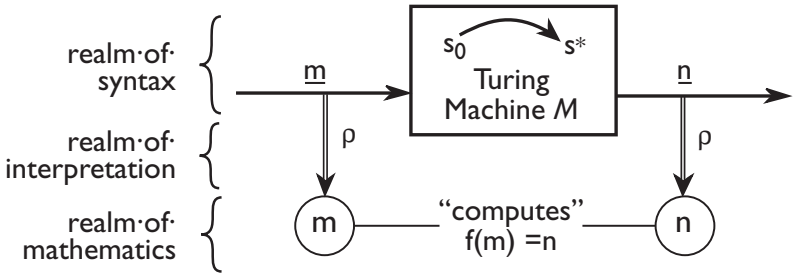

Figure 3 — Turing machines

5. But as we saw in the first critique, issues of what is inside, and what is outside, can strongly influence analysis. So will need to keep an eye on this distinction.

B. Marks, alphabets, and languages

1. Squares can be marked or blank, in the usual way
2. Strictly speaking, controller can only deal with a single square at a time
3. But for most purposes, it is *connected* (adjacent) *sequences of marks* that are important
   a. I.e., in terms of what "problem" the machine solves, sequences play the critical role
   b. Units of semantical interpretation
4. Here, therefore, I will use (underlined) variables **m**, **n** to mean **mark sequences**
5. I.e., will treat **m**, **n** as something like *formulae* in a specified *language*
6. This enlargement of the subject matter will be relatively harmless (for our purposes here)
7. We just need to be careful that controller isn't asked to maintain an indefinite amount of internal state
8. Conceptually, too, down the road (we'll get to this in a few weeks) there will be a difference between *marks* and *configurations of marks* ("configurations" aren't as obviously concrete). At that point, it may be important to distinguish between the two.

C. Sets of inputs

1. Can define a Turing machine to give specific output, given a specific input
2. But that is boring
3. What one always does is to define a (one) machine to work over *sets* of possible inputs
4. That's why what a Turing machine (controller) is identified with a *function*, not simply a single pairing of input and output.

D. Vocabulary

1. We need to be careful in how we speak about these machines
2. There are two ways to use the word 'compute'[8]
   a. **Opaquely:** Like 'utter': to refer to *marks* (⇐ most computer scientists)
   b. **Transparently:** Like 'describe': to refer to *numbers* (⇐ many logicians, mathematicians)
3. I don't have a judgment that one is right, one is wrong
4. We just need to be careful
5. N.B.: I will use **input** and **output** to signify *marks*. I.e., will use 'input' and 'output' *opaquely*

[8]See the sidebar "Two readings of 'compute' " on page III·1·22 and III·1·23 of *AOS·III*.

E. Representation

1. Take tapes to be *representational*
2. E.g., unary or binary numerals
3. Cf. morals from first construal
   a. Machine is to run in virtue of **potent** (syntactic, positive, effective) properties of marks
   b. Opens up all the standard questions of *what properties are effective, potent*, etc.
   c. Whereas in first critique we couldn't take them on, here we (finally) will
4. However: there are *non*-parallels, between the two construals, on issues of interpretation
   a. Domain of interpretation
      i. Typically mathematical (numbers and functions)
      ii. Much more constrained than one normally thinks of, in FSM
   b. Potency predicate: *effectiveness* (*efficacy*), rather than *syntax* (why? what about semantics? is there an independence assumption in the background?)
   c. Representational arrangements of marks often called *encodings*
      i. Narrower term than 'representation' (especially than 'description')
      ii. But still: encoding is an *intentional* (semantic) *directedness relation*
      iii. So keeping an eye on it will be tremendously important
   d. General point (◆)
      i. Whereas designation and denotation are typically treated (i.e., in traditional formal models of logic, language, etc.) as *strict, non-transitive*, and hence as **theoretically opaque** (in the sense that it is important to distinguish between something that denotes and what it denotes, and between something that designates and what it designates),
      ii. Models or encodings, in contrast, are treated as **theoretically transparent** (i.e., the entities at either end can typically be identified, the modeling relation is generally taken to be transitive, etc.[9])
5. We will want to keep an eye on these things, to see whether
   a. They are just gratuitous differences in vocabulary, from the discourse of FSM
   b. Or, whether they reflect some deeper as-yet unarticulated difference
6. I will ultimately argue that they *do* reflect a deeper difference—a difference of enormous importance for our project.
7. Note (in passing) that we interpret not only *marks*, but *machines themselves* (at least associate them with the mathematical functions that they "compute"). More on this presently.

F. (Negative) role of theory

1. It is often said that you don't need a *theory* of computing for positive results: to show that something is computable. To do that, you simply show how to compute that thing.
2. Rather, the theory is needed (people say) for negative results: to show that something *cannot* be computed
3. This may be true, for general theoretical purposes, but it won't work for a foundational investigation

---

[9] In the sense that if $\mathbf{m}_1$ is a model of $\mathbf{m}_2$, and $\mathbf{m}_2$ is a model of $\mathbf{m}_3$, then $\mathbf{m}_1$ is taken to be a model of $\mathbf{m}_3$ as well.

4. Example: function from $\underline{\mathbf{n}} \Rightarrow \mathbf{n}$
   a. This may be a compelling function, but it is not entirely clear what it is an example of.
   b. On a representational theory of mind, what is most likely is that the function from $\underline{\mathbf{n}} \Rightarrow \mathbf{n}$ is *transparent*. Why?
   c. Because:
      i. If we represent $\underline{\mathbf{n}}$ as '$\underline{\mathbf{n}}$', and $\mathbf{n}$ as '$\mathbf{n}$' (i.e., as $\underline{\mathbf{n}}$) …
      ii. Then understanding $\underline{\mathbf{n}} \Rightarrow \mathbf{n}$ simply involves disquotation
         α. Which, as we saw in the first critique, is easy
      iii. And *disquotation*, from '$\underline{\mathbf{n}}$' ⇒ '$\mathbf{n}$', surely *is* effective (as we also saw in first critique)
   d. But it doesn't follow that the original function, $\underline{\mathbf{n}} \Rightarrow \mathbf{n}$, is effective after all! (In fact it is not going to be clear what it is to assert—indeed, whether it may not be a category error— that the function $\underline{\mathbf{n}} \Rightarrow \mathbf{n}$ is effective).
5. So we need a foundational reconstruction even to understand the simplest examples
6. This is all "par for the course," for the foundationalist

G. In sum (six points):
1. Take "Turing machine" to denote the controller, not the whole controller plus tape (and especially not the controller plus *marked* tape;
2. Take $\underline{\mathbf{m}}$, $\underline{\mathbf{n}}$, etc. as (meta-theoretic) variables ranging over mark sequences (configurations of marks)
3. Define machines (i.e., controllers) to operate over different *sets* of inputs
4. Distinguish opaque (like "utter") or transparent (like "describe") uses of 'compute.'
5. Take marks or mark sequences (i.e., configurations of marks) as denoting (numbers, usually)
6. Try to discern what the theory says it *is* to compute something, positively; don't just rely on it for negative results (what *can't* be computed)

H. Now turn to the two more substantial issues:
1. Representation / encoding (this time)
2. Basic notion of, and intuitions behind, effectiveness (on Thursday)

IV. Representation / Encoding

A. Logic: has a double subject matter
1. Syntax, derivability, etc., very much on the table ⇐ cf. proof theory
2. Semantics, interpretation, etc., *also* on the table ⇐ cf. model theory
3. Moreover, the *bite* of a logical system has to do with how the two relate (this is what we have taken as the primary dialectic underwriting our entire understanding of computing)

B. Recursion theory, computability theory, etc.
1. Seem to deal with *only a single subject matter*
2. We say, for example, that products of large primes are hard to factor, or that propositional satisfiability is NP-complete.

C. What happens to the encoding?
1. The coding is universally agreed (and admitted) to be important
2. *Everyone* says (on page 4): must use a **reasonable encoding**
3. One question we will focus on will be: what a reasonable encoding is

4. Examples:
   a. Base π arithmetic (cf. AOS·III·1, page III·1·21)
   b. What symbols are allowed (e.g., square-root)
5. But rarely (as far as I know) no analysis of what a reasonable encoding is
6. Doesn't figure as prominently in our understanding of what is going on

D. Some other *non-parallels* between Turing machines and FSM
   1. Domain of interpretation
      a. Typically mathematical (numbers and functions)
      b. Much more constrained than one normally thinks of, in FSM
         i. Especially in our case, because we read 'symbol' so widely
      c. Also much more *uniform:* talk in terms of it, directly
   2. Potency predicate: *effectiveness* (≡ *efficacy*), rather than *syntax*
   3. Representational arrangements of marks often called *encodings*
      a. Narrower term than 'representation' (especially than 'description')
   4. Do these differences mean anything?

E. This whole situation is odd. Think about the structure of the investigation
   1. Issues of representation, interpretation, semantics, etc.—i.e., legitimacy of codes, conditions and constraints on interpretation functions ρ, etc.—are *of absolutely central concern*;
   2. They are also (as is widely admitted) *essential to the recursion-theoretic story*; and yet (in spite of both these facts)
   3. They do not figure centrally in resulting substantive claims.

F. We should post a very serious flag, to understand what is going on (◆)

G. Specifically, we need to know
   1. What a reasonable encoding is
   2. Why, in spite of its evident importance, the issue is accorded secondary status, in the way things are normally presented; and
   3. Whether, once we have answered the first two questions, an appropriate response would be to put the issue of what constitutes a reasonable encoding more squarely on the table, in full theoretic view.

H. Hint (promise, in fact) of what is to come:
   1. *Yes,* we will figure out what a reasonable encoding is
   2. *Yes,* we will figure out why it has been accorded secondary status
   3. *No,* we won't argue that it should be on the table, first-class part of the subject matter.
   4. Rather, we will say that practice was right (all along) to treat it in a secondary fashion
   5. In fact I will recommend moving it even *further out of explicit subject matter view* (rather than bringing it more explicitly into view).
   6. On the other hand, the cost of this move is high

V. Effectiveness

A. We will turn to the intuitions behind the notion of effectiveness next time (i.e., next Tuesday!).

—— *end of file* ——