A Theory of Marks

I. Preliminaries

A. Review

1. Last time, we introduced ★—a machine alleged to solve the halting problem.
2. Except of course that was false: ★ doesn't *really* solve the halting problem. It was a cheat.
3. All ★ did was to meet a series of *formal characterisations* of the halting problem.
4. In each case, we took the fact that ★ met the formal requirements we had laid out to be evidence that our characterisations were inadequately formulated—prompting us to revise them. This led to a succession of increasingly strong versions.

B. Effectively discriminable paths

1. The basic intuition we came to was that any machine that genuinely solved the halting problem should meet 3 conditions:

> **C1** *Different* inputs should lead to the *same* output, if they represent the *same* halting behaviour;
>
> **C2** *Different* inputs should lead to *different* outputs, if they represent *different* halting behaviour; and
>
> **C3** Counterfactually, any given input $\underline{\mathbf{m}}_i$, $\underline{\mathbf{n}}_i$ *should have led to a different output*, had the (metaphysical, ontological, conceptual, whatever) facts about whether $\mathfrak{M}_i$ halts on $\underline{\mathbf{n}}_{ij}$ been different.[1]

2. The problem was that ★ met this condition already, if one allows "same" and "different" to be interpreted as "represents 0" and "represents 1," respectively.
3. That led us to need to adopt *effective* version of the constraints (figure 1, on the next page):

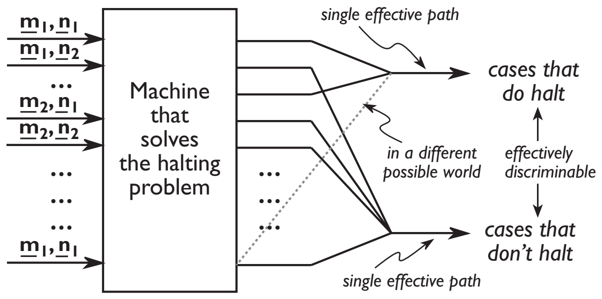
> **P3** One must be able to marshal all the inputs that represent situations where machines halt (i.e., that represent 0) onto *one effective path*, and similarly to marshal all the inputs that represent situations where machines do not halt (i.e., that represent 1) onto *a different effective path*.

4. That in turn led to a new formulation of the problem:

[1]For a discussion of what it is to say that the output of a methamtically-defined machine "might have been different" see footnote #4 on page 8·16 of the notes for lecture 8b.

H4 Given as input marks <u>**m**</u> and <u>**n**</u>, representing the numbers **m** and **n**, respectively, produce as output marks <u>**0**</u> or <u>**1**</u>, representing 0 or 1, respectively, depending on whether the Turing machine $\mathfrak{M}$ modeled by the set of quintuples μ coded by the number **m** would or would not halt, if given as input the mark <u>**n'**</u> modeled by the number **n**, such that (i) all tokens of <u>**0**</u> lead (immediately?) to a single effective state or path, (ii) all tokens of <u>**1**</u> lead (immediately) to a single effective state or path, and (iii) all tokens of <u>**0**</u> are (again, immediately) effectively discriminable from all tokens of <u>**1**</u>.

0. This worked: **H4** was strong enough to "catch" ★. That is, ★ does not meet **H4**.
6. In general, the goal of **H4** was to force the solution to meet the condition that the following all be *effectively done*:
   a. *Differentiate* all <u>**0**</u>s from all <u>**1**</u>s;
   b. *Unify* all <u>**0**</u>s; and
   c. *Unify* all <u>**1**</u>s.
7. It is this triple demand—when understood that *all three criteria must be met effectively*—that seems to be infeasible.



$\underline{\mathbf{m}}_1, \underline{\mathbf{n}}_1$
$\underline{\mathbf{m}}_1, \underline{\mathbf{n}}_2$
...
$\underline{\mathbf{m}}_2, \underline{\mathbf{n}}_1$
$\underline{\mathbf{m}}_2, \underline{\mathbf{n}}_2$
...
$\underline{\mathbf{m}}_1, \underline{\mathbf{n}}_1$
Machine that solves the halting problem
single effective path
cases that do halt
in a different possible world
effectively discriminable
cases that don't halt
single effective path

Figure 1 — Effectively discriminable paths

C. Status
1. Are we done? Is **H4** okay?
2. No. There are at least two major problems (which we started talking about last week):
   a. Circularity: The most glaring problem is that **H4** is *defined in terms of the very notion of "effectiveness" we are supposed to be defining*. So if **H4** is the right constraint (in terms of which to specify the conditions on a possible answer), then it appears that we are going to have to define a notion of effectiveness in a way that is explanatorily prior to its use in defining Turing machines. That (as we will see) is a *very substantial demand*.
   b. Coding: We got into the issue of ★ by using a non-standard encoding, after a long discussion of the strong, necessary conditions on the interpretation function ρ (from marks to numbers).[2] But the proposed solution ★, captured in constraint **H4**, doesn't mention representation or coding at all. So something strange is going on. In particular: we haven't yet answered our driving question: of what the constraints on ρ must be, in order to formulate an adequate Turing-machine theoretic conception of computing.
2. We will pursue both of these issues more in a moment.
3. But first we need to note something crucial: *the generality of the problem we have uncovered.*

B. Generality
1. Nothing in the conditions manifested in **H4** is specific to the halting problem. The same issue would come up with respect to any problem at all, such as multiplication, addition, etc.

[2]We argued that ρ must be *more* restrictive than the effective computability relation $\phi_{\text{mathematical}}$; in contrast to the interpretation function for English, necessary to understand the halting problem as the halting problem, which we saw had (in at least a certain sense) to be *less* restrictive than $\phi_{\text{mathematical}}$.

2. In order to prevent ★-like "cheats, *every* "problem" to be solved by Turing machines will need reformulation in **H4**-like terms (or whatever better version we end up selecting).
3. Multiplication
   a. For example, suppose we were to try to define a machine $\mathfrak{M}^*$ to multiply integers— i.e., to compute the multiplication function.
   b. As in the case of the halting problem, it is insufficient to define $\mathfrak{M}^*$ *purely* in terms of FAVs: i.e., given as input numbers $\mathbf{m}_1$ and $\mathbf{m}_2$, produce as output that number $\mathbf{m}$ such that $\mathbf{m} = \mathbf{m}_1 \cdot \mathbf{m}_2$. What is needed is a definition formulated in terms of marks.
   c. As before, the naïve suggestion would be to require that $\mathfrak{M}^*$, given as input representations (marks) $\underline{\mathbf{m}_1}$ and $\underline{\mathbf{m}_2}$ of numbers $\mathbf{m}_1$ and $\mathbf{m}_2$, respectively, produce as output mark $\underline{\mathbf{m}}$ such that $\underline{\mathbf{m}}$ represents $\mathbf{m}_1 \cdot \mathbf{m}_2$.
   . But this could be solved by a machine as vacuous as ★—in particular, by one that simply prints out (as its output) "$\underline{\mathbf{m}_1 \cdot \mathbf{m}_2}$". Obviously, on a very simple interpretation function ρ, $\underline{\mathbf{m}_1 \cdot \mathbf{m}_2}$ designates the product of $\mathbf{m}$ and $\mathbf{n}$.[3]
   e. Instead, therefore, to "force" $\mathbf{m}_1 \cdot \mathbf{m}_2$ actually to be *computed*, we need either to specify a specific (constrained) encoding relation ρ—or in general to say something like:

> **MULT** Given as input marks $\underline{\mathbf{m}_1}$ and $\underline{\mathbf{m}_2}$, representing numbers $\mathbf{m}_1$ and $\mathbf{m}_2$, respectively, produce as output mark $\underline{\mathbf{m}}$, representing $\mathbf{m}_1 \cdot \mathbf{m}_2$ such that, for any given (product) number $\mathbf{m}$, all tokens of $\underline{\mathbf{m}}$ lead (immediately?) to a single effective path or state.[4]

   f. I.e., as in the previous example, the idea is to "collect" similar outputs into a single effective path, as in figure 1. But there is a remaining worry: it is not clearly that **MULT** is strong enough to count as *multiplying*. We will get back to that later.
4. Since similar considerations show that *all* Turing-machine problems are vulnerable to this kind of manoeuvre. Indeed:

> ◆ All individuations, discriminations, and typings of Turing machines—distinguishing one state from another, distinguishing one machine from another, etc.—are subject to the same sorts of (**H4**-type) arguments we have used against ★.

5. As a result, we should build these **H4**-like constraints into the very notion of the machine.

C. So that's our current goal
   1. To figure out how to state the constraints *in general*, in a way that is applicable to all Turing-machine problems.
   2. Once we have done that, we will be able to assess the consequences for the entire Turing-theoretic conception of computation.

[3]Cf. the machine we talked about, which represented numbers with (i.e., used numerals that were) sequences of binary representations of the number's prime factors. This machine could multiple simply by concatenation.
[4]Again, a similar condition would need to be stated for the inputs—in this and all the examples we are considering.

II. Uninterpreted marks

A. Representation

1. The moral to which all these examples lead—something we already knew—is very general:

> **P4** No requirement formulated at the level of functions, arguments, and values can ever force a function to be "computed."

2. This is because (I claim) computation is not something that *happens* at the level of FAVs.
3. That is not to say that computation (or its inputs and outputs) is not *about* FAVs. To that extent, the mathematical reading can stand.
4. But there is a very general claim being advanced:

> **P5** Computing a function does not "happen" at the level of FAVs. It happens at the level at which functions, arguments, and values are *represented*.

B. Uninterpreted marks

1. There is one more piece to the puzzle (which will allow us finally to discharge the circular definition of effectiveness that we ended up with).
2. Non-interpretation
   a. Note that neither of the outputs specified in **H4** any longer have *any representational requirements placed on them*
   b. Literally, the specification claims that they need to be of the form **<u>0</u>** or **<u>1</u>**, representing 0 or 1, respectively. But *no work is done by having this requirement.*
   c. Everything that *matters* in **H4** ended up being specified in the "single effective path" part, which refers only to the outputs as (uninterpreted) marks.
3. We can thus formulate a revised and final version of the halting problem that gets rid of this excess representational baggage:

> **H5** Given as input marks **<u>m</u>** and **<u>n</u>**, representing the numbers **m** and **n**, respectively, produce as output marks β or γ, depending on whether the Turing machine $\mathfrak{M}$ modeled by the set of quintuples μ coded by the number **m** would or would not halt, if given as input the mark **<u>n'</u>** modeled by the number **n**, such that (i) all tokens of β lead (immediately?) to a single effective state or path, (ii) all tokens of γ lead (immediately) to a single effective state or path, and (iii) all tokens of β are (again, immediately) effectively discriminable from all tokens of γ.

C. This is the final formulation we will lay out here.

1. Note: there is an open question: about what the right conditions would be on the *input* side.
2. Rather than examining that issue now, however, we will proceed onto the positive story—and then, once it is in place, come back and revisit the question of what constitutes a legitimate *problem* (input) for the halting problem—or, indeed, for any problem that we want to be "solved" by a Turing machine.

II. Turning the representation relation upside down

A. Consider the argument so far
   1. We started out considering FAVs, but not marks. But that failed: it wasn't strong enough (to catch ★).
   2. We moved to considering FAVs, marks, *and their interpretation* ρ (thus answering—in the negative—question **Q3b**[5]). That helped, but ultimately ρ snuck back in, so as to defeat us.
   3. So we got rid of ρ, and ended up considering marks alone, not FAVs at all.
   4. That is: we started at the bottom (of our "three-storey diagram"), moved upwards, and when we got to the top, forgot about the bottom.

B. What about ρ?
   1. We started out terribly concerned about constraining ρ (we've had discussions about that)
   2. We also noted that, in practice ("reasonable encodings"), ρ is *severely constrained*
      a. It is *much* more constrained than general semantic interpretation—and even more constrained that the interpretation functions for even simple logical languages
      b. Reasonable encodings are typically *one-to-one*, sustaining neither ambiguity nor coreference (equality)—a terribly strong and (in general) unrealistic restriction
   3. We also noted that semantics in general, and entire issue of what constitutes a reasonable encoding, is curiously absent from recursion / computability theory as typically formulated.
   4. Now—in at least our chosen problem—interpretation has disappeared!
   5. I.e., reasonable encodings are
      a. Simple
      b. Ubiquitous
      c. Untheorised (almost invisible—remember the "unitary focus")
      d. Irrelevant in at least our example case

C. What are we to conclude from all that? Here's a hint:
   0. We said that the outputs of the machine we were considering didn't need to be interpreted as marks **0** or **1** representing 0 or 1, respectively, but could instead be taken to be arbitrary (distinct) mark (types) β or γ.
   2. The choice of those names β and γ was of course arbitrary.
   3. So why didn't we follow standard convention for classifying binary oppositions, and simply associate the two groups with the numbers 0 and 1?
   4. This suggestion opens the door to the final, crucial insight.

> **P5** The marks **0** and **1** do not represent the numbers 0 and 1, after all. Rather, the number 0 and 1 represent (at least model) the marks **0** and **1**!

D. Status
   1. Conclusion P5 is incredibly consequential—as we will be seeing, over the coming weeks.
   2. Once we've opened the door, we're have to take down the frame and remove the hinges!

[5]See Figure 2 on page 8·12 of lecture notes 8b.

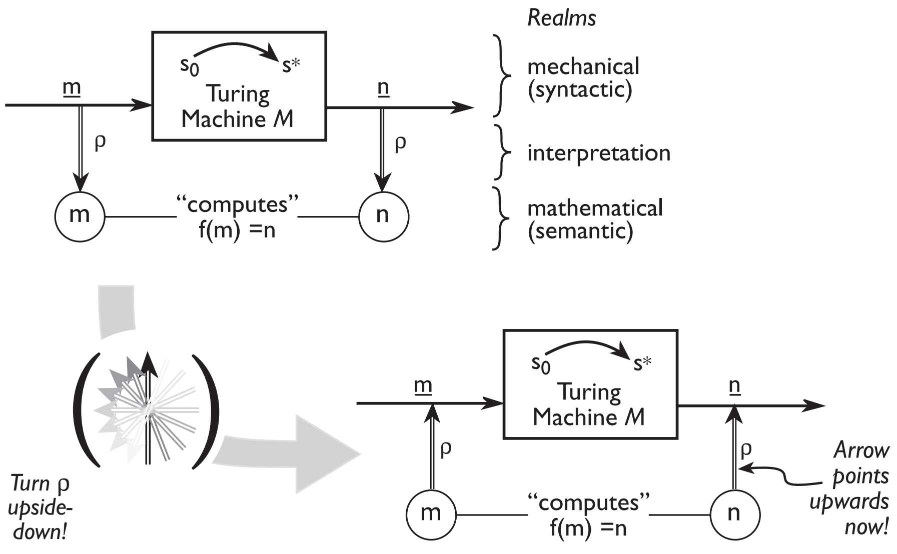


Figure 2—From marks representing numbers to numbers representing marks

III. A theory of marks

A. Conclusion

1. We have arrived at the situation indicated in figure 2.
0. What is needed is to recognise a shift: *the representation relation* ρ *runs the other way around!*
3. The "real" situation is revealed to be entirely analogous to what we saw in classical physics!
4. This explains a myriad things
   a. Why ρ had to be extremely simple, usually one-to-one (and why it must be massively simpler than general human interpretation).
      i. In a way, the answer is simple: *it's because what really needs to be true of* ρ *is that it be able to be inverted!*

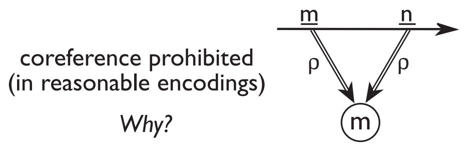


Figure 3a—Coreference prohibited?

      ii. Thus consider *co-reference* (i.e., equality), an absolutely ubiquitous semantical relationship. Why must it be prohibited, in reasonable encodings (figure 3a)?
      iii. Answer: if, as we're claiming, the real representation relation runs the other way around, co-reference would be *ambiguity*—which is *obviously* something you don't want, in your theoretical model (figure 3b).
   b. Why ρ is never theorized.
      i. It is never theorized because *it is not part of the subject matter.*

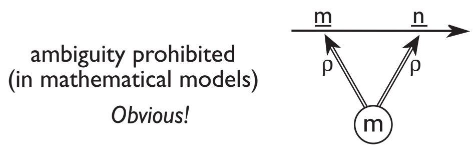


Figure 3b—Ambiguity prohibited!

      ii. Rather, it is part of the theorist's analytic or explanatory equipment (just as mathematics is part of the theoretic or explanatory equipment of any modern science).
      iii. What is required, in such a case, is just what happens, in practice: people need to *agree* on ρ (as part of their theoretic practice), without needing to construct a *the-*

ory of ρ (and more than scientists are in general required to deliver theories of the meta-language in which the results of their theories are presented).

c. It explains why recursion and complexity theory have a unitary subject matter.

i. In discussing the differences between logic and recursion/computability theory, we said that in logic one focuses on *two* things—syntactic relations (such as derivability) and semantic relations (such as entailment), as well as on the relations between them—whereas in recursion/computability theory one focuses on only *one* thing.

ii. If (as we're claiming) the interpretation of the marks is part of the theoretical modeling, not part of the subject matter, it becomes obvious why this is: it is because there really *is* only a single, unified subject matter (mathematically modeled).

2. These things are all explained because computability theory is in the end like any other mathematical theory. For any number of ordinary theoretical purposes, it turns out to be useful to indirectly classify one set of (physical) phenomena in terms of various (abstract) mathematical entities, in the centuries-long tradition of the arithmeticisation of science.

B. Reconstruction

1. What about the real subject matter of the theory of effective computability?

2. Computability theory—complexity theory, the theory of effective computability (maybe even recursion theory?)—was never interested in representation.

3. Rather, computability theory is a **theory of marks**.

a. Not of marks in all their concrete or semantical glory, of course (as we've seen, marks can exemplify all sorts of properties, not least including semantic ones, to which the theory of effective computability is blind).

b. Better: computability theory is a theory of *the effective properties of marks*.

c. But even that is not quite right, since it is not so much interested in them statically, as it is in the kinds of (potentially dynamic) processes of effective transformation that can be defined over them.

d. Moreover, it is not interested them in detail, in their multiplicity of weights, sizes, colours, and shapes.

e. Rather it is interested in them at a higher level of constitutive abstraction.

C. Summary

1. To put it all together, we will say that what is traditionally goes by the name "the theory of computation"—including complexity theory, the theory of computability, and perhaps recursion theory as well—has nothing to do with representation, semantics, or intentionality.

P6 The "theory of effective computation" *is not a theory of computation at all.*

2. Remarks

a. NB: there still *is* computing.

b. It remains an essentially interpreted phenomenon (as we have claimed since the beginning). It is certainly likely (tautologous?) that people "compute" the answers to various mathematical queries.

c. It is even possible that Turing machines compute—or that (physical) Turing machines can be used as an (informal) model of what it *is* to compute. None of these things is being denied.

d. All that is being said is that the received theory of effective computability is not a theory of any of them.

e. Rather, it is a theory of the *constraints of effectiveness* to which all representational computational—which is to say, all computing—is subject.

3. Put that in a single phrase:

> **P7** What has been called a theory of "effective computability" is in fact (in spite of the press) a **mathematical theory of the flow of effect**.

4. That's why we said, at the beginning, that the "theory of effective computability" got one word right, out of two. It's a theory of the *effective*.

II. Consequences

A. Admit: **P7** is a very strong claim

B. Next time

1. Talk more about why we should believe it
2. Also: what the consequences of believing it would be.

—— *end of file* ——