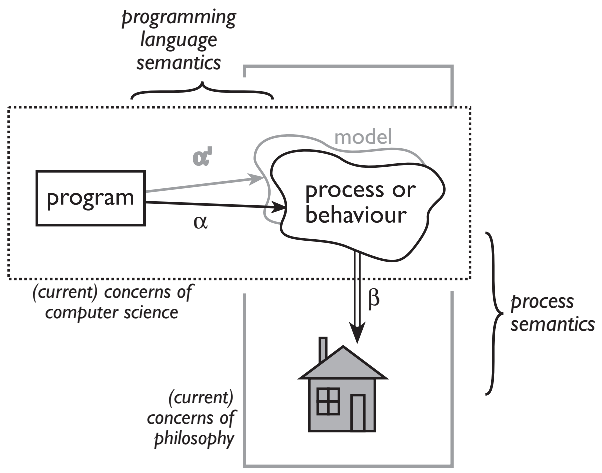
Towards a Mathematical Theory of Effectiveness

I. Preliminaries

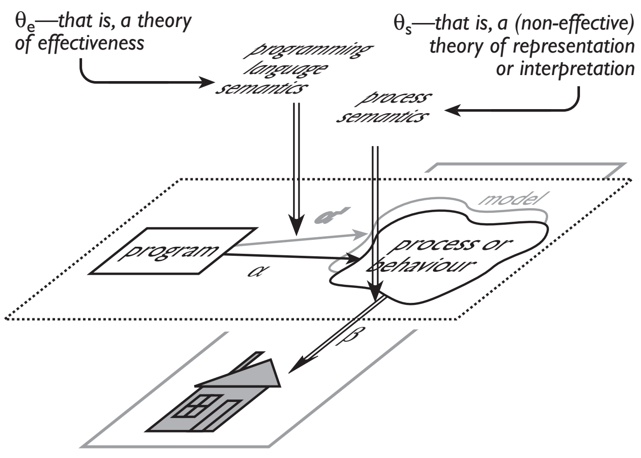
A. Review

1. This is our final day analysing the second construal (the theory of effective computability)!
2. As we said Tuesday, the basic picture we're working with is given in figure 1, but a better depiction is given in figure 2, reflecting the double aspect of the change we are recommending:
   a. The relation between "marks" (and other effective structures) and numbers (and other mathematical entities) has been turned around, implying that the mathematical entities represent (model) the marks, rather than the marks representing (encoding or denoting) the mathematical entities; and
   b. The inverted relation is taken to be a relation between the theory and the subject matter, rather than being subject-matter-internal.



programming language semantics
model
α'
program
α
process or behaviour
(current) concerns of computer science
β
process semantics
(current) concerns of philosophy

Figure 1 — Program and process semantics



θ$_e$—that is, a theory of effectiveness
programming language semantics
θ$_s$—that is, a (non-effective) theory of representation or interpretation
process semantics
model
α'
program
α
process or behaviour
β

Figure 2 — Theory, program, and subject matter

B. Plan

1. So far, we've argued that effectiveness should be understood neither semantically nor abstractly, but concretely—i.e., as a physical notion
2. What we haven't done is to see what would be involved in **taking effectiveness as physical**
3. That is: we have recommended a reconstruction of the theory of computability (including recommending that it be renamed as a theory of effectiveness or a theory of causality, since by our lights it does not count as a theory of *computing*), but we haven't indicated

what a positive reconstruction would look like.

4. Today we'll indicate what would be involved in doing that.
5. We'll proceed in two steps or "transformations" (in terms of figure 3b):
   a. **First transformation:** strengthen the (vertical) connection between "effectiveness" and physicality
   b. **Second transformation:** loosen the alleged (horizontal) connection between "effectiveness" and semantics or intentionality (since it doesn't really deal with it).
6. Note: this transformation will make EC—reconstructed as a theory of effectiveness—more like a positive reading of FSM (figure 3a)!
7. In doing this, we make sense of the move we made earlier,[1] in moving the "positive" reading of "formal" (in the first construal) to the EC reconstruction was right! (figure 4).

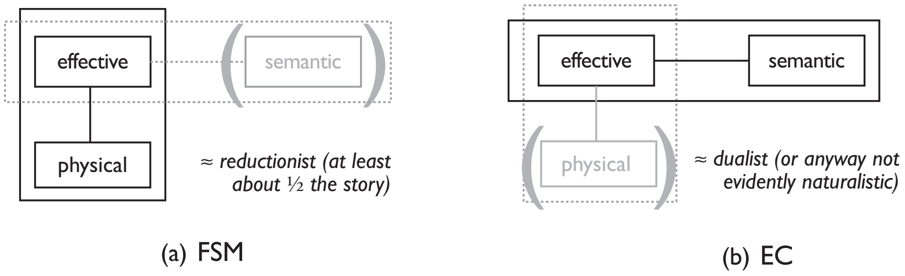

effective
semantic
physical
≈ reductionist (at least about ½ the story)
(a) FSM
effective
semantic
physical
≈ dualist (or anyway not evidently naturalistic)
(b) EC

Figure 3 — Conceptual Structure of the Two Construals

C. Notes
   1. Today, we can only gesture towards the kinds of work that would be involved in each of the two transformations. As we will see, it would take an enormous amount of work to do them properly.
   2. Also, in line with this week being one of summary and distillation, we will be repeating—but gathering together—a number of remarks that have been made in passing, over the past several weeks.

II. First transformation

A. Introduction
   1. The goal of the first transformation is to show what it would be to take the subject matter (of the reconstructed EC theory) to be *concrete*, rather than abstract.
   2. I.e., just as we *denied formality* in the reconstructive phase of the first critique, so too here we will explore the consequences of denying formality in the reconstruction of the second construal.

B. Preliminary arguments
   1. As a way of loosening the territory up, one would want to

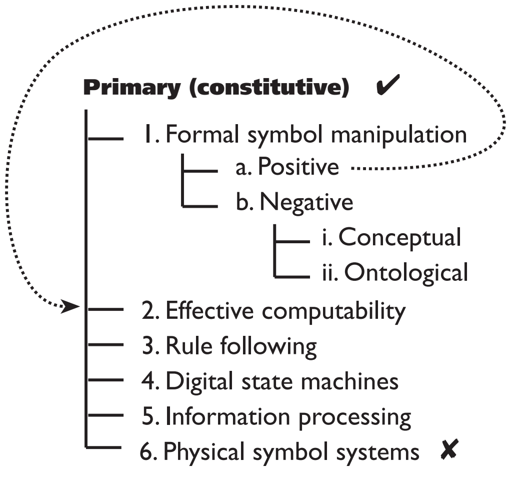

Primary (constitutive) ✔
1. Formal symbol manipulation
a. Positive
b. Negative
i. Conceptual
ii. Ontological
2. Effective computability
3. Rule following
4. Digital state machines
5. Information processing
6. Physical symbol systems ✘

Figure 4 — Positive reading of 'formality' (in the first construal)

[1]See the discussion associated with figure 1 on page 4·2 of lecture notes 4a (Jan 30).

   a. Review arguments that have been advanced for taking computation or computability (i.e., the subject matter of the theory at hand) to be abstract.
   b. Suggest informal reasons why it may not be abstract

2. **Con-abstraction:** arguments about multiple realisability.
   a. Based on a confusion on two readings of 'abstract'
      i. *Treating things at (different) levels of abstraction*
      ii. *Treating something independent of any material facts of embodiment at all*
   b. There is no doubt that we want to do (i). But that doesn't imply (ii).
      i. Yes, we can implement an adder out of silicon, Lincoln logs, or by storing the bits in standing resonance waves in elongated Coke bottles
      ii. But one can also build hospitals out of wood, concrete, adobe, and probably Coke bottles as well.
   c. As we have said several times, by now, that doesn't make hospitals—or computers—abstract (in a Platonic sense)
   d. That is: we should recognise that:

◆ Although any given physical system can be analysed at (many) different levels of abstraction, to analyse a system at a (higher) level of abstraction doesn't require or entail *taking it to be an abstract object*.

3. **Pro-concreteness:** Argument that our notion of effectiveness is tied to physical realisability
   a. In a way, our entire argument about the machine ★, and the inexorable importance of marks, is a "pro-concreteness" argument. But one can argue to that effect from other directions, as well.[2]
   b. Suppose, for example (contrary to all expectation), that we were discover a microphysical particle that solved the travelling salesman problem in unit time.
   c. Whenever this particle—call it a **travelon**—is within a certain critical distance of a group of **n** other particles, it accelerates in such a way that the first **n** derivatives of its position, with respect to time, in order, taken as vectors, point towards those particles in the order of the shortest trip that visited them all.[3]
   d. Assume that problems of measurement and set-up are sufficiently tractable that devices exploiting the travelon's behaviour can be marketed at modest cost.
   e. If such a particle were discovered, would we want the theory of effective computation to continue to claim that theoretically optimal solutions of travelling salesman problems require exponential time?

[2]The point of ★ was not so much simply to argue for the concreteness of computing, as to show that what matters about computing lies in the realm of the marks, not in the realm of what the marks denote.

[3]By assuming that the magnitudes of the travelon's generalised acceleration vectors, which are not needed for the "solution," tend towards zero whenever the neighbouring particles' respective positions approach a configuration where the shortest path visiting them all change discontinuously, it is even possible to preserve various forms of continuity.

f. Surely not. That is: for anything remotely resembling current computer science, the answer is *no*. On the contrary, conferences would spring up to thrash out the *new accounts of what could now be done in what space-time envelopes.*

g. If this is true, computability must not be abstract

C. Project

1. Basic point would be to take computation to be a concrete, occurrent phenomenon
2. One strategy for doing this (we mentioned this in an earlier class) would be to recognise that when one talks of computations as being abstract, one would recognise that one was talking about the **abstract type of a concrete phenomenon**..
   a. But since all types are abstract (in that sense), it would be better described in the way we have described it before: as a **concrete type**.
   b. This would open the way for direct ties between recursion-theoretic tradition and the foundations of computing and physics.
3. Requirements on such a theory
   a. Reconceiving computation in terms of concrete types shouldn't leave one's understanding intact. Rather, it should be the first enabling step in a move to open up a whole spate of other questions, which could then be addressed (and answered).
   b. For example, it should allow one to explain *why* impossible ("non-computable") behaviours are impossible, rather than just stating *that* they are impossible—perhaps by showing that they violate conservation laws, or require an infinite amount of energy to accomplish in finite time, or contradict the laws of quantum mechanics.
   c. Or perhaps do *not* contradict the laws of quantum mechanics.
   d. Instead, show that Turing-computability is not only *computationally* classical, but classical in its *physics* as well—raising the possibility that "quantum computers" could outstrip the traditionally-conceived limits.
   e. Or perhaps (see part IV) show that the limits stem from digitality: that a continuous notion of effectiveness can be defined to handle turbulence, basins of attraction, and other features of non-linear dynamics that outstrips the current computability limits.[4]
4. General characteristics
   a. Get rid of strange predicates (like Searle's "wall" processor, "grue," etc.)
   b. Make way for temporal properties (real-time, rhythm, etc.) to be taken seriously:
      i. Examples:
         α. Pure temporal properties: *moments*, *durations*, *intervals*;
         β. Units: *minutes*, *seconds*, *nanoseconds*
         χ. Relational predicates: *before*, *during*, *after*, *simultaneously*
         δ. Rhythms: *cycles*, *oscillations*, *ringing*, *quiescence*
         ε. Complex measures: *operations per second*, *megaflops*
         φ. And so on and so forth.

[4]This is of course already being done. But note that it is described using traditional language (e.g., that a continuous or turbulent computer can "compute more or different functions"). On second transformation (below) this language will have to change.

ii. Such notions should be part of a physicalised theory
iii. I.e., we would want to develop a proper *temporal theory of process*
iv. Cf. real-time languages
v. Also: deal explicitly with *performance*
vi. Would "doing the same thing" be maintainable across such a change?

c. 3-D packing: part of the theory of parallel systems
d. Explain various intellectual ties:
i. Between information and quantum mechanics (Landauer, Dallas conferences, "physics and computation" communities, ties between information loss and heat, etc.)
ii. With dynamical systems, SFI, etc.
e. Take software to be concrete

5. In a way, all these steps, which I am characterising here as the "simplest" consequences of reconceiving of computing as concrete, are starting to happen. But they are just the tip of the iceberg.

D. Universality (example of adopting this "concrete" viewpoint)

1. For tougher consequences, consider the fundamental notion of (computational) universality.
2. I.e., Consider what happens when one proves that a universal machine **U** can "do the same thing" as some other machine **M**. Such equivalences are shown in the usual way: one writes a program $\mathbf{p}_M$ to implement **M** in **U**, so that, given some input α on which **M** would output β, the pair $\{\mathbf{p}_M, \alpha\}$, if submitted to **U**, would—though probably a long time later—yield β.
3. Question: what's equivalent to **M**? Classically: **U**. But what makes that a reasonable answer?
4. If **U** is taken to be the *controller*, then the claim is false (even on coarse-grained non-temporal metrics).
5. Rather, what is equivalent to **M**—the only thing that can reasonably be considered to manifest **M**-like behaviour—is the *combination of controller* **U** *and program* $\mathbf{p}_M$.
6. So why do we say that **U** is "as powerful as **M**"? Because of classical distinction between hardware and software (or state of memory).
7. What is really going on is this:

a. Given program $\mathbf{p}_m$, understood by prior agreement to be a configured arrangement of the physical world; and controller **U**, also understood to be a configured arrangement of the physical world; and
b. And recognising that
i. Program $\mathbf{p}_m$ is not the slightest bit general or "universal-like", but on the contrary is very specifically targeted at **M**;
ii. The complexity profile of the combined system will in all likelihood be on the order of 99% $\mathbf{p}_m$, 1% **U**; and
iii. The governing metric of equivalence is very abstract—setting aside, among other things, all real-time temporal conditions on **M**'s behaviour
c. Then two conclusions emerge:
i. *It is odd to attribute the universality to what is by far the smaller part of the combination.*
ii. It is not the *same* machine that "compute the same function," after all; it is a *larger* machine—potentially *much* larger.

iii. It no longer seems to be so impressive that one can construct such a configuration, given that there are virtually no constraints on $\mathbf{p}_M$.

d. In fact the overall result seems to lessen to the following:

◆ **Universality (I):** *Certain* arrangements of the physical world α (universal controllers) have the property that, when conjoined in appropriate ways with arbitrarily large, complex *other* configurations of the physical world β, yield systems which can simulate the behaviour of *any* configuration of the physical world γ, with respect to an arbitrarily abstract isomorphism between the inputs and outputs of one (γ) and the inputs and outputs of the other ($\alpha+\beta$), so long as one ignores the temporal dimension of both machines.

e. Admittedly, normally this difference: only α is a source of *energy* or *anima*; the program (β) is taken to be static or at least passive.

f. I.e., the universal machine is the *motor*.

g. Leads to another restatement:

◆ **Universality (II):** Given (a) an erector set (or equivalent stock of compositionally-assemblable parts, and (b) a motor, it is possible to build a device—of potentially Rube-Goldberg complexity (no matter)—whose surface behaviour is isomorphic to that of any other device you can build, so long as one ignores the fact that the simulating device is liable to run arbitrarily slowly.

8. Somehow this doesn't seem so surprising anymore. Is that all there is to it?
9. Question
   a. Is there anything that these caricatures are missing?
   b. If so, what is it?
   c. If not, shouldn't we start describing the universality results in this way?
   d. If so, then does it mean all the hoopla over the 'universal computing' results last century was distraction—merely a symptom of the state of conceptual confusion?

III. Second transformation

A. Status

1. This last discussion of universality has already started on the second transformation: ridding the vocabulary of the theory (of effectiveness) of intentional vocabulary.
2. Remainder of today: a glimpse of the sorts of reformulation that would be implied
3. Some relatively straightforward:
   a. Cf. Kolmogorov information: rename ⇒ something like *order* or *orderliness*
   b. Cf. decidability of logic: can no longer call a problem "**decidable**," since system isn't *deciding* anything (decision is semantic, interpreted notion). Instead: **canonicalise**?
4. Look at four somewhat more complicated examples
5. Admit at the outset:

a. I really don't know the answers to these questions

b. So everything from here on (rest of Part III) has the status of wonder and speculation.

B. Halting problem

1. Would have to be reformulated, again without semantic vocabulary

2. Cf. discussions of why certain problems are unsolvable, normally phrased in terms of self-reference (self-representation), etc. This is not a diagnosis we can say, anymore; since *reference* and *representation* are verboten. Replace with something like "structural similarity"?

3. Possible suggestion (merest hint):

a. Maybe what is going on is that any device of finite size can be "swamped" by an input whose complexity drowns its own—so that it can no longer keep track of what is happening. If this is so, then what must matter, in the proof of the unsolvability of the problem, has something to do with the *similarity of the effective structure of the input and the effective structure of the device's internal states*. Or if not *similarity*, then *vulnerability* of the device's internal states to a particular form of input.

b. For example, imagine the following game.

i. One person is supposed to come up with a machine, **H**, such that, given a pair of inputs, {**m**, **n**}, **H** should halt just in case the machine "effectively isomorphic" to **m** would halt if given **n** as input.

ii. Your job, given a candidate **H**, is to "fool" it, by coming up with two pairs of inputs, {$\mathbf{m}_1$, $\mathbf{n}_1$} and {$\mathbf{m}_2$, $\mathbf{n}_2$}, such that the machine that is "effectively isomorphic"[5] to $\mathbf{m}_1$ halts on $\mathbf{n}_1$, and the machine that is isomorphic to $\mathbf{m}_2$ does not halt on $\mathbf{n}_2$, but such that **H** cannot tell the difference (i.e., **H** will either halt on both, or fail to halt on both, thereby failing to meet its initial mandate).

c. Note that on these formulations—both of the machine **H**, and of the pair of inputs that "fool" it—*there is no worry about representation or semantic interpretation.*

d. Rather, your strategy would be to devise inputs {$\mathbf{m}_1$, $\mathbf{n}_1$} and {$\mathbf{m}_2$, $\mathbf{n}_2$} that would "break" **H**, in the sense of driving it to overwrite its memory, or forget where it was coming from, or otherwise confuse it, so that it cannot maintain the differences between them.

e. I.e., you would arrange to have the effective path that **H** follows, when {$\mathbf{m}_1$, $\mathbf{n}_1$} is the input, *merge* with the effective path it follows when {$\mathbf{m}_2$, $\mathbf{n}_2$} is the input. Once they merged, **H** would never be able to pull them apart again.

f. Doing this would have to do with the relationship between the effective structure of the inputs and the effective or causal pathways inside **H**. *Interpretation* would be irrelevant.

g. The structure of a successful strategy would be interesting. Would it rely on sheer complexity, for example, as implied by the metaphor of "drowning"? Or is a more specific style of self-similarity necessary to defeat any candidate? Suppose you had an architectural diagram of the effective structure **e** of a device, and wanted to generate two sets of inputs {$\mathbf{m}_1$, $\mathbf{n}_1$} and {$\mathbf{m}_2$, $\mathbf{n}_2$} that would defeat it. Can anything interesting be said about the function ***f*** from **e** to {{$\mathbf{m}_1$, $\mathbf{n}_1$}, {$\mathbf{m}_2$, $\mathbf{n}_2$}}? These are the sorts of question to which the proposed reconception should be held accountable.

C. Computing functions

[5]Rather than "denoted or modeled by"!

1. What would we say about a machine to, say, do multiplication (call it **M***).
2. Setup
   a. If we take the traditional relation between marks and numbers as one of *reverse classification*, all that is being said about **M***, when we describe it as multiplying numbers, is that it moves from an effective input configuration classifiable with a pair of numbers onto an effective output configuration classifiable by their product.
   b. Nothing has been claimed about whether these configurations are semantically interpretable (though of course nothing *precludes* their being semantically interpretable).[6]
   c. Take 'numeral' to designate *whatever representations meet the conditions we are after*—so that we can say that a system multiplies two numbers **m** and **n**, if, given as input numerals **<u>m</u>** and **<u>n</u>** representing **m** and **n**, respectively, it produces as output numeral **<u>a</u>** representing their product **m · n**.
   d. This is terminologically compact, but so far represents no progress; the question has simply been deflected on one of what counts as a numeral.
3. What distinguishes the "appropriate" choice of notations?
4. Some candidates (what distinguishes these choices?)
   a. Traditional notations: using familiar positional notations (binary, decimal, etc.), so that to multiply fourteen by fifteen is to take in '14' and '15' and produce '210', or to take in '1110' and '1111' and produce '11010010'.
   b. Two notations that are *outside* the normal scheme.[7]
      i. The base-π notation discussed earlier (so that 3 + 1 = 10.220122…, π^2 = 100, etc.)[8]
      ii. A notation system that represents numbers by an ascending series of their prime factors, where those factors are in turn represented in unary. Thus fourteen and fifteen would be represented by '11·1111111' and '111·11111', respectively; their product, by '11·111·11111·1111111'.
   c. Roman numerals. They share with the second scheme the property that larger numbers are sometimes represented by shorter codes
   d. The simplest scheme of all: *unary*.

[6]As usual, just because (the new version of) the story does not describe **M*** *as* multiplying, that does not imply that **M*** is not *in fact* multiplying.

[7]The complexity profiles of standard operations on these last two coding schemes would be non-standard. E.g., on the first scheme the circumference of a circle of diameter **d** could be "calculated" with complete accuracy simply by shifting **d** left one place, whereas integer addition could take an indefinite amount of time, depending on the accuracy demanded, and in general could never be performed perfectly. On the second scheme, similarly, prime factorisation, which on standard schemes is so difficult as to be used as the foundation for all modern encryption schemes, would be trivial; multiplication would be almost as easy (linear in the length of the codes); but addition would again be excruciating slow.

[8]Four digits would suffice—'0', '1', '2', and '3', though numerals would not be unique (10 = 3.01102111…). On the other hand numerals are not unique in decimal notation either; every finite numeral denotes the same number as the infinite number of other numerals that are alike except with one or more preceding 0s—as well as the (single) infinite numeral that is identical to the original except that its last non-zero digit d is replaced by d-1 followed by an infinite string of 9s.

5. Some intuitions:
   a. The prime factorisation scheme somehow seems furthest from counting as a numeral, at least in the ways in which we ordinarily deal with numbers.
   b. The problems with integer addition in the π-based and prime factorisation schemes seems severe—a fact that should count heavily against them.
   c. It nonetheless seems that the π-based scheme, though not in general very useful, might be an appropriate representational system for dealing with geometry, especially with the area of circles. (I.e., though it is a little hard to imagine, it seems plausible that in an especially curvaceous universe, among a race of creatures who did not count much, but for whom areas of circles and volumes of spheres were an important currency of interaction—and thus of intuitive mathematical practice—the π-based system might serve well, perhaps even better than our normal one.)
   d. Unary seems crucial. It is not a trivial fact that unary almost invariably serves as the ground floor case of arithmetic representation: not only for Turing machines, but also for Peano arithmetic and λ-calculus numerals (both of the latter, in the standard notations, use **n** instances of a syntactic structure to represent the number **n**).
6. What is the moral? A suggestion:
   a. To count as a numeral, a representation should *enable simple effective access to an exemplification of the represented cardinality.*
   b. This ties straight back to lessons learned from counting, in the first critique. It also explains why unary numerals are central: because they *exemplify the cardinality they represent* (for unary it is a short effective route from numeral to number). Finally, though we neither use unary numerals in practice, nor go on using them in theory for very long, because the relation between number and numeral is *too* close for practicality, we do not let the relation stray very far, either.
   c. Presumably the procedure for moving from **n** entities to the numeral $\underline{\mathbf{n}}$ must be approximately linear; that for moving from $\underline{\mathbf{n}}$ to $\underline{\mathbf{n{\cdot}m}}$ require no more than approximately **m** steps (including the specific case of 1 step when **m** is 1), etc.[9]
   d. Note two (non-incidental) facts
      i. This would be an operation of the sort that in the first critique we identified as *crossing a semantic boundary*.
      ii. The proposal makes sense of an intuition that surfaced, quite a while ago, when we were trying to tie down the halting problem: namely, that the relation between marks and their referents itself be effective.[10]

[9]I say "approximately" because of the obvious fact that adding 1 to 999,999 takes 7 steps, not 1, but I believe that the underlying idea about effective participatory access is still sound. «reference analogous problems with CD players?»

[10]At that point we claimed that this intuition was incoherent; have we changed our mind? No, for two reasons. First, we were then looking for a notion of effectiveness, and claimed that it could not be defined, within the framework of Turing machine calculation, with reference to the mark–number relation. By now we are no longer asking for a definition of effectiveness; that has been deferred to the underlying physics. Second, it remains true that the relation between marks and numbers in the Turing-theoretic conception is an inverse one of methodo-

D. Complexity
   1. What are we to make of complexity theory?
   2. It is about *the complexity of transforming one effective configuration of the world into another.*
      a. I.e., given a device effectively arranged in one way, how many steps or adjustments in structure will it have to go through in order to end up in some other configuration?
      b. In and of itself, the answer to this has nothing to say about (nor is it vulnerable to what anyone else might say about) how or even whether that structure is interpreted.

E. Compositionality
   1. It is often pointed out that one of the most characteristic features of semantical accounts is their *compositionality*: the fact that the "semantic value" of a complex structure is a function of the "semantic value" of its parts.
   2. How much of that intuition is genuinely semantical?
   3. The new theory of effectiveness or mechanism should apply as readily to motors and Meccano as it does to programs and Pascal. Do they have compositional structure?
   4. In a way, *yes*. It is a commonplace of engineering, after all, that the behaviour of a complex whole (especially of complex artifacts) is a function of the behaviour of its parts, too.
      a. E.g., consider a Meccano set, consisting of girders, wheels, plates, axles, pulleys, etc.
      b. These can be combined in a variety of ways: a continuous number of ways, given the fastening technology, but no matter.
      c. It would presumably be straightforward to define a mathematical model for each piece type, and mathematical operators corresponding to ways of fastening them together, so that the structure of behaviour of a complex artefact made out of the pieces was mathematically derivable.[11]
      d. I.e., a *denotational semantics for Meccano.*
   5. Cf. MEMS: this is a positive result, since microminiaturisation is quickly making silicon wafers into as hospitable environments for motors as for logic gates.

IV. Conclusion

A. Carrying out these two transformations would be an enormous amount of work

B. I believe, though, that what we have already seen suggests that the work would be worthwhile

C. Two primary benefits: the positive reconstruction should:
   1. Lead to a deeper understanding of things that are genuinely physical / material
   2. Open the way for a theory that *really* deals with the other half of our fundamental dialect: semantics. (◆)

—— *end of file* ——

---

logical classification. The isomorphism between that situation and the one described here explains why that situation is so often (though incorrectly) viewed as description of multiplication. Isomorphism is not identity (as usual it is context-dependence that tears the two apart).

[11]It is especially easy to imagine for statics: the model of each piece might simply be the volume of 3-space it occupied, plus perhaps some indication of its structural strength. But with a little imagination the same could be done for dynamics, so that the extensional behaviour of a pressure gauge could be derived an analysis of the (extensional) behaviour of each of its pieces, given an account of how they were assembled.