Computing Machines and the Philosophy of Mind

CSHPM Notes September 2023 (Vol. 55, No. 4)

Jessie Hall (Institute for the History and Philosophy of Science and Technology, University of Toronto)

CSHPM Notes brings scholarly work on the history and philosophy of mathematics to the broader mathematics community. Authors are members of the Canadian Society for History and Philosophy of Mathematics (CSHPM). Comments and suggestions are welcome; they may be directed to either of the column's co-editors:

Amy Ackerberg-Hastings, *independent scholar (aackerbe@verizon.net)*
Hardy Grant, *York University [retired] (hardygrant@yahoo.com)*

Artificial intelligence and cognitive science seem to share a common assumption: that computers and minds might be similar in some respect or other. Whatever respect this might be, the notion of artificial *intelligence* takes it that a computer *could be like* a mind in the sense that the cognitive capacity of *intelligence* could be attributed to an artificial computing machine. Cognitive science, or more specifically, computational theories of mind (CTM), take it that the mind *is like* a computer, in that the mind—thought, or more scientifically, cognition——is constituted of computational processes performed by the brain. These might seem like perfectly intelligible claims, but they are so only if it is clear what it is to be a computer (or, what it is for a physical system to compute). For the last few years, I have been preoccupied with the revelation that, in fact, what it is to be a computer is far from clear. The effort to make sense of this unclarity has spanned the philosophy of mind, philosophy of language, and philosophy of mathematics.

In the recent philosophical literature on the subject, a variety of attempts have been undertaken to make sense of computers as *physical machines* [e.g**.**, 6; 8; 4; 9; 3]. The conception of a physical machine at hand is heavily influenced by the 21st-century tradition of the 'new mechanical philosophy' or *new mechanism*. New mechanism takes machines to be not just those mechanisms built by human beings for a variety of purposes, but *any* physical system whose behaviour is understood as a (sometimes exceedingly complex) arrangement of parts and subparts causally related to one another and couched within a broader causal nexus.

A preliminary hope for any account of physical computing machines (what I refer to alternatively as 'computers') is that the account coheres with our intuitions about the machines we already *call* 'computers'. Unfortunately, intuitions do not always agree. For instance, among such machines, some but perhaps not all cognitive scientists count brains. Whether we count brains as computing machines or not, the relevant question to ask is: What must a physical machine be like, to be called a computer? One of the most frequently encountered answers takes the form of what is often called a 'mapping' account of computation: A physical machine is a computer if it implements a computable function, and a machine implements a computable function if there is a *mapping* between that machine and the function.

It is a sensible-sounding view, especially in light of the fact that the concept of a computable function—computability itself—is most famously articulated in terms of a machine, the Turing machine (TM). This Turing machine is named for Alan Turing (1912–1954, Figure 1), the British mathematician who first described the machine in 1936—in his terms, the automatic machine or 'a-machine'. The TM was a preliminary formalism for addressing a vexing foundational problem in mathematics, familiar to contemporary computer scientists as the 'halting problem'. Turing machines are not *physical* machines, but *notional* ones. It would be an infelicitous leap to conflate the a-machine described in [10; 11] with the physical 'machines' as construed above. It is a substantial question to ask: *When does* a physical machine implement a Turing machine? [12] The mapping view says that a physical machine *P* implements a TM when there is a mapping from *P* to the TM.

Figure 1. Alan Turing in 1935. *Wikimedia Commons*, CC BY-SA 4.0.

Unfortunately, this basic picture has led to significant and highly unintuitive consequences. The philosopher Hilary Putnam (1926–2016, Figure 2) has shown that under this simple mapping account of computation—one that identifies computation with the implementation of computable functions, and implementation with a mapping from machine to automata—*every ordinary open system implements every finite state automaton* (FSA). Loosely, an open system is any system where matter and energy can traverse the boundary of the system—most, if not all meso-scale physical mechanisms are of this sort (for instance, a granite rock might have a clear enough boundary, but its formation, cooling, and erosion constitute matter changes, while e.g. the heat of the sun impinges on its mean kinetic energy). Finite state automata are a subclass of Turing machines—a subclass of computable functions. In other words, every (open) meso-scale physical system is a computer, under the simple mapping account of computation [7, Appendix]. This thesis is called 'pancomputationalism', and the version articulated by Putnam is one of the strongest; every system computes *every* FSA. Other philosophers [see, for example, 1; 2] admit only a more limited version of pancomputationalism: every physical system computes *at least one* FSA.

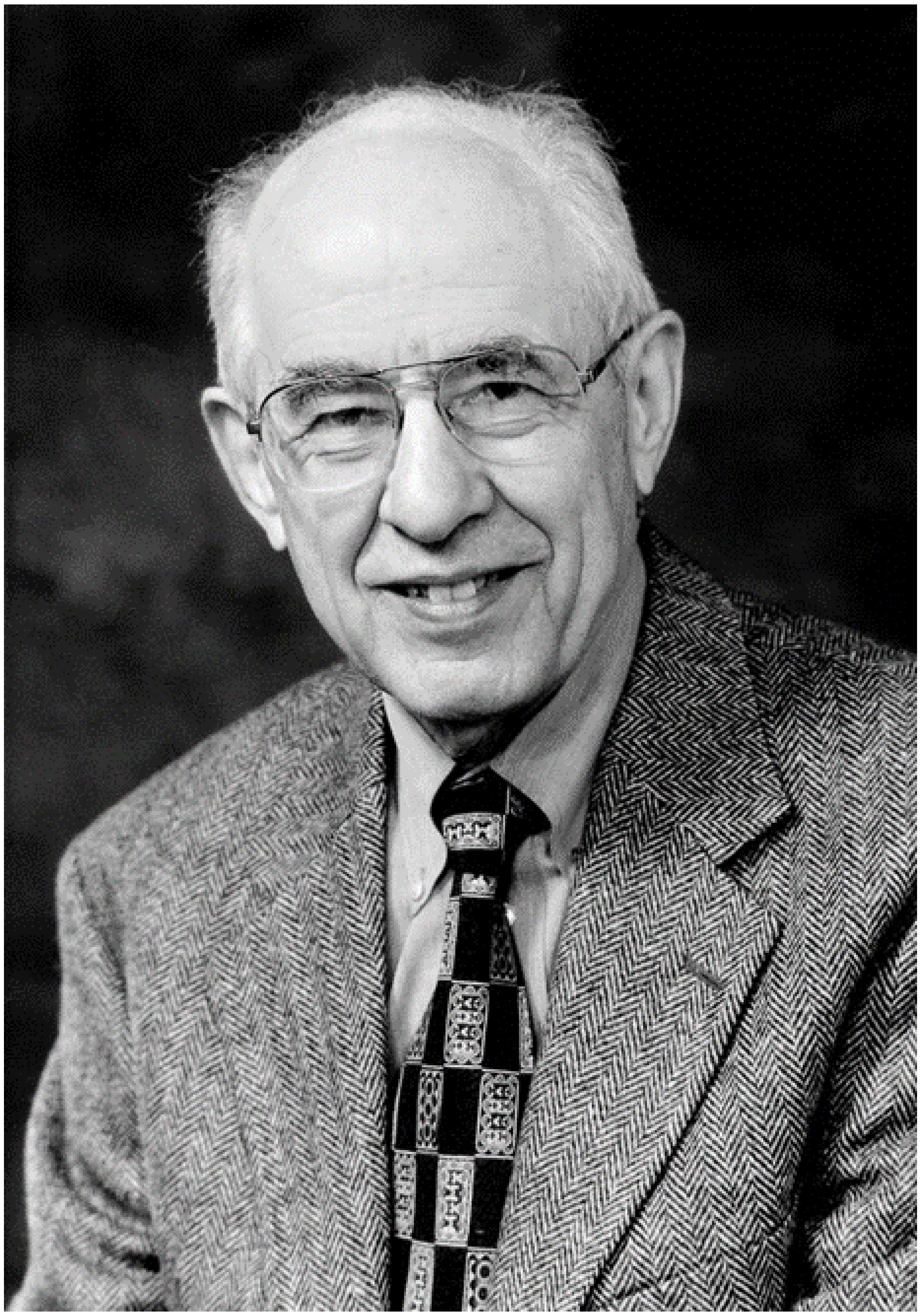


Figure 2. Hilary Putnam in 2006. *Wikimedia Commons*, CC BY-SA 2.5.

The way in which pancomputationalism arises under these 'mapping' views of computing is a consequence of at least one of its basic commitments. Mapping accounts are composed of three basic ingredients:

1. Automata; (models of) the species of expressions captured by Turing computability (TMs and FSA are the preferred formalisms);
2. The implementation relation, which involves a mapping from physical system to automaton model;
3. What is often hidden from view but essential: a model of the physical system that is to be mapped to the automaton—the source domain of the mapping from physical machine *P* to automaton *A*.

One popular strategy for avoiding Putnam's unlimited pancomputationalism is to constrain (2) by placing conditions on those implementations considered to be legitimate. The strategy has revealed an opposition between those who believe that computation essentially involves semantic properties, and those who believe that semantic properties are not essential to computation, that computers can be individuated without appeal to semantic properties. Under the semantic view, the only legitimate implementations are mappings from physical machine *P* to automaton *A* that meet certain representational constraints. Under non-semantic views, physical machine *P* implements automaton *A* only when the automaton to which the physical system is mapped meets any one, or a combination, of the following:

- The mapping between *P* and *A* supports counterfactuals regarding *P*'s behaviour;
- *A* reflects the causal organization of *P*;
- *A* respects and/or mirrors scientific explanation of the behaviour of *P*.

Defenders of the latter view seem to be motivated by the conviction that computing machines must be naturalizable, by which they usually mean explicable in terms of, or even reducible to, physical properties, their causal interactions, and organization. Semantic properties, on the other hand, seem resistant to explication or reduction of this sort, and thus threaten the kind of naturalizability pursued by new mechanists. Typically, strategies of the non-semantic sort make two assumptions. First, that there is one model of a given physical system (3), and second, that pancomputationalism arises primarily from spurious mappings from that model (3) to automata (1), mappings which are spurious in virtue of failing to preserve the mechanistic properties of the system given by the model (3). This strategy has been unsuccessful at evading pancomputationalism, and, in my view, unsuccessful in virtue of the assumption regarding (3). The problem does not arise from spurious mappings from a given model of a physical system to many different automata, but because the physical system itself can be modelled in a variety of ways, all of which map to (are in fact isomorphic with) automata that fulfill the non-semantic implementation constraints.

One might suppose that there is a *correct*, and maybe even a *singular* correct model of a physical system, which is given by a scientific theory of that system. Unfortunately, real science does not bear out that supposition. For example, the mechanism of ATP production in cellular respiration might be explained in terms of macronutrients and chemical compounds (e.g., Krebs cycle), or alternatively, in terms of the biokinematics of motor proteins (e.g. the myosin 'walk').

For a simpler illustration of this idea, I've devised an example of a pendulum (Figure 3). We can just as easily talk about the states of a pendulum given by thermodynamics—e.g., the mass P heating up and cooling down under the sun—as we can the states of the pendulum given by kinematics, and even a kinematic treatment of the pendulum can produce a multitude of models. The pendulum shown in Figure 3 might have its behaviour described in a variety of ways. One way might be to define the states of the pendulum in terms of the velocity of mass P, and define the inputs in terms of the force F_P on P (a function of Θ).

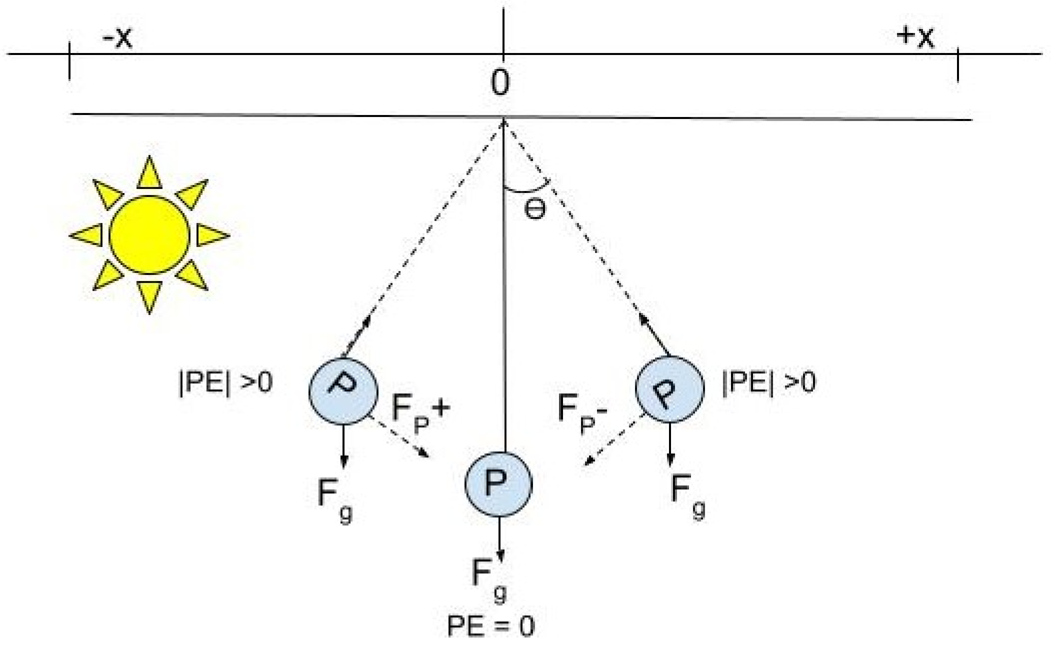
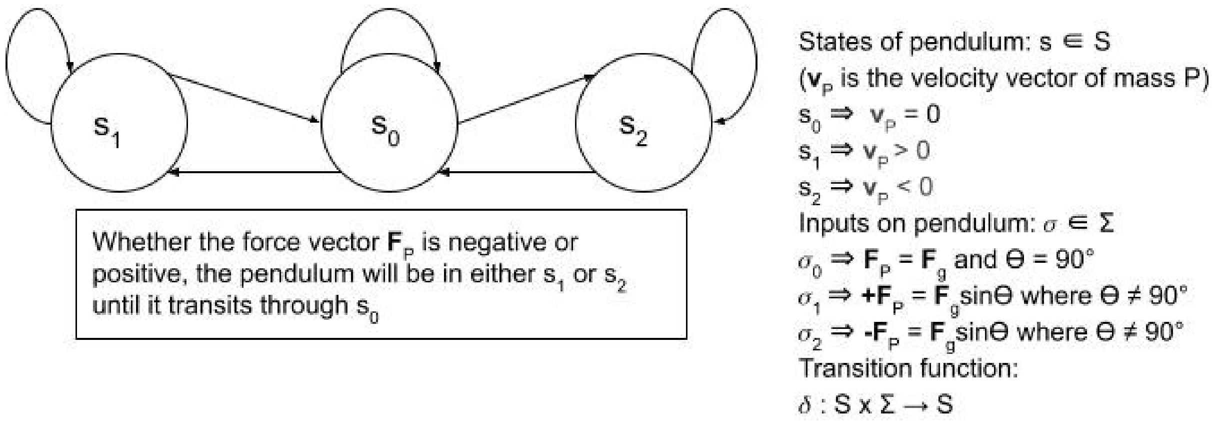


Figure 3. States of a pendulum. Diagram created by author.

Alternatively, the pendulum's states might just as well be defined in terms of the potential energy (PE) of mass P, and the inputs defined in terms of positions along the x-axis. Yet another model of the pendulum might define its states in terms of the relative temperature of mass P as a function of time, and the inputs as the angles of the sun relative to mass P. Figure 4 illustrates a model of the pendulum behaviour under the first description.



Whether the force vector $\mathbf{F}_P$ is negative or positive, the pendulum will be in either s_1 or s_2 until it transits through s_0

States of pendulum: $s \in S$
($\mathbf{v}_P$ is the velocity vector of mass P)
$s_0 \Rightarrow v_P = 0$
$s_1 \Rightarrow v_P > 0$
$s_2 \Rightarrow v_P < 0$
Inputs on pendulum: $\sigma \in \Sigma$
$\sigma_0 \Rightarrow \mathbf{F}_P = \mathbf{F}_g$ and $\Theta = 90°$
$\sigma_1 \Rightarrow +\mathbf{F}_P = \mathbf{F}_g \sin\Theta$ where $\Theta \neq 90°$
$\sigma_2 \Rightarrow -\mathbf{F}_P = \mathbf{F}_g \sin\Theta$ where $\Theta \neq 90°$
Transition function:
$\delta : S \times \Sigma \rightarrow S$

Figure 4. Alternate model of states of a pendulum. Diagram created by author.

While opinions may diverge on whether a pendulum ought to count as a computer, unfortunately such models—which conform to scientific theories pertinent to the behaviour of a mechanism, and support counterfactuals regarding possible transitions—can be devised for *any* mechanism. One and the same mechanism can have a variety of behaviours amenable to a multitude of models, and any given (causally explicated, counterfactual-supporting) behaviour can be modelled in a variety of ways. The result is that any given machine implements—that is, maps to—a variety of automata, simply in virtue of being a machine. Unless we are prepared to capitulate to pancomputationalism, a better account of computers must be formulated.

The alternatives found in the semanticist camp—those who take semantic properties to be essential to the individuation of computing machines—have a different set of problems (which may very well include pancomputationalism). Chief among them is that it is not entirely clear what it means to be individuated by semantic properties, nor indeed, what a semantic property even is. 'Semantic properties' might be construed as *relations* between expressions and their extensions (e.g., reference, synonymy, truth), but they might also be construed as whatever it is that is taken to *fix* those relations. Theorization over both construals of semantics constitutes a vast and contested literature.

Supposing this problem can be addressed, there is still the problem of naturalization. On the one hand, one might reject a notion of naturalization that conforms to the strictures of new mechanism—by rejecting the notion that everything 'natural' is reducible to or explicable in terms of physical properties, their causal relations, and organization. On the other hand, either the semantic theory undergirding the individuation of computers must itself be (mechanistically) naturalizable, or it must be accepted that computers are simply not naturalizable.

One might, at this point, simply wish to give up on explaining what a computer is. Unfortunately, as suggested at the outset, there are stakes to giving up. If computing in physical systems remains unaccounted for, then claims that certain physical systems, such as brains, are computers, are claims without substance. Likewise for claims that certain machines, in virtue of their computational properties, could be intelligent; their subject matter is a mere mirage.

There are two possible outcomes from here. In one scenario, the approaches just detailed are hopelessly misguided, but out there somewhere is a correct account of computers that does not suffer from any of the problems just canvassed. In the other scenario, one or more of the problems above persists—pancomputationalism, (non-)naturalizability—in which case our claims about brains and AI must be reassessed: If everything computes, what does it mean to say a brain does? If every mechanism computes, and computation in a machine is sufficient for intelligence, then does every mechanism have (some form of) intelligence? If computation is semantic, can it still be naturalized? And so on. These are among the questions that will be waiting in the wings for any account of computing machines.

Note: The subject of the present essay is classified in the Mathematics Subject Classification System (MSC2020) under the headings of 'computer science', 'theory of computing', and 'artificial intelligence'. In older versions of MSC the heading 'logic in the philosophy of science' also appears. Some of the sources below are reviewed in *Mathematical Reviews* (*MR*), and for other sources closely-related publications by the authors are reviewed. The *MR* identifier is provided below for sources that are reviewed in *MR*.

References

[1] Chalmers, D. (1994) On implementing a computation. *Minds and Machines* 4, 391–402.

[2] Chalmers D. (1996) Does a rock implement every finite state automaton? *Synthese* 108, 310–333. See also review 1412784 in *MR*.

[3] Coehlo Mollo, D. (2020) Against computational perspectivalism. *The British Journal for the Philosophy of Science* 72(4), 1129–1153.

[4] Dewhurst, J. (2018) Computing Mechanisms without Proper Functions. *Minds and Machines* 28, 569–588.

[5] Gandy, R. (1980) Church's Thesis and Principles for Mechanisms. *Studies in Logic and the Foundations of Mathematics* 101**,** 123–148.

[6] Piccinini, G. (2015) *Physical Computation: A Mechanistic Account*. Oxford: Oxford University Press. Various articles by Piccinini related to the subject of this book are reviewed in *MR*.

[7] Putnam, H. (1988) *Representation and Reality*. Cambridge: MIT Press. See also review 0958690 in *MR*.

[8] Shagrir, O. (2018) In defense of the semantic view of computation. *Synthese* 197(9), 4083–4108. See also review 4136266 in *MR*.

[9] Schweizer, P. (2019) Computation in Physical Systems: A Normative Mapping Account. In *On the Cognitive, Ethical, and Scientific Dimensions of Artificial Intelligence: Themes from IACAP 2016*, edited by D. Berkich and M. V. d'Alfonso, 27–47. Berlin: Springer.

[10] Turing, A.M. (1937) On computable numbers with an application to the Entscheidungsproblem. *Proceedings of the London Mathematical Society*, ser. 2, 42(1),230–265. See also review 1577030 in *MR*.

[11] Turing, A. M. (1938) On computable numbers, with an application to the Entscheidungsproblem. A correction. *Proceedings of the London Mathematical Society*, ser. 2, 43(1) 544–546.

[12] It is important here to note two senses of the term "Turing machine". In the broad sense, "Turing machine" refers to the a-machine described in Turing (1937) consisting of a read/write head that can move left and right, and read and write symbols, and an infinite tape, divided into squares, upon which the read/write head writes and reads symbols. In the narrower sense, a 'Turing machine' might refer to a 'machine description' which specifies the (finite) list of steps that the a-machine follows for producing a particular sequence.

Jessie Hall is a doctoral candidate at the Institute for the History and Philosophy of Science and Technology at the University of Toronto. Her research examines the influences of mathematical (Turing) computability, functionalism, and various stripes of reductionism on conceptions of physical computational systems, brains as computing systems, and 'abstract' computing.