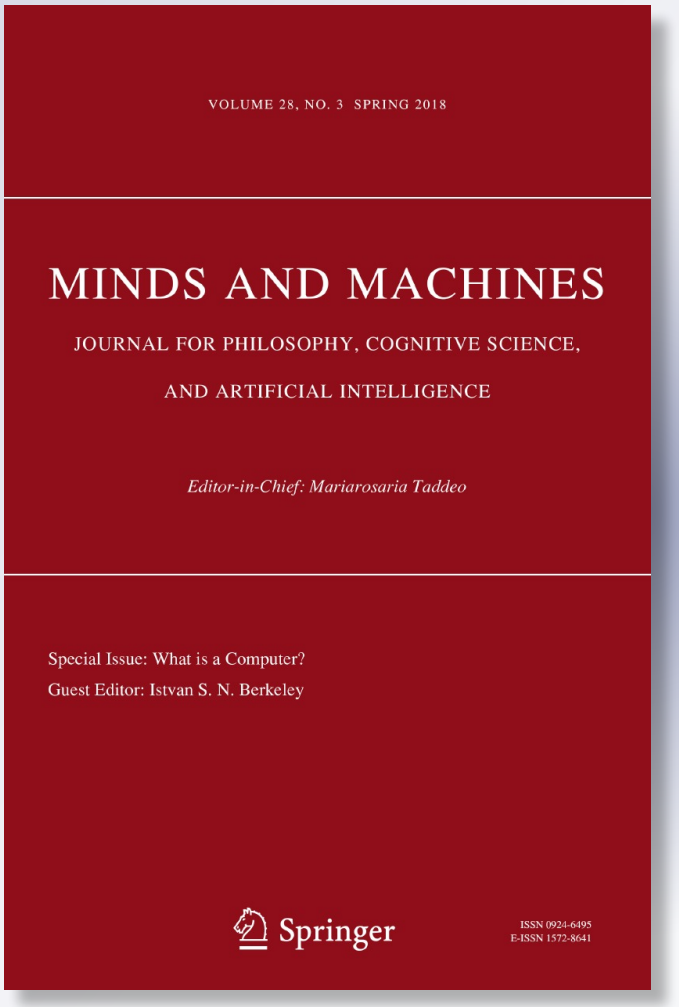
What is a Computer? A Survey

William J. Rapaport

Minds and Machines
Journal for Artificial Intelligence,
Philosophy and Cognitive Science

ISSN 0924-6495
Volume 28
Number 3

Minds & Machines (2018) 28:385-426
DOI 10.1007/s11023-018-9465-6



Springer

Minds & Machines (2018) 28:385–426
https://doi.org/10.1007/s11023-018-9465-6

What is a Computer? A Survey

William J. Rapaport[1,2,3]

Received: 28 December 2017 / Accepted: 15 May 2018 / Published online: 25 May 2018

Abstract A critical survey of some attempts to define 'computer', beginning with some informal ones (from reference books, and definitions due to H. Simon, A.L. Samuel, and M. Davis), then critically evaluating those of three philosophers (J.R. Searle, P.J. Hayes, and G. Piccinini), and concluding with an examination of whether the brain and the universe are computers.

Keywords Computers · Computation · Herbert Simon · Arthur Samuel · Martin Davis · John Searle · Patrick Hayes · Gualtiero Piccinini · Turing machines

In virtue of what is something a computer? Why do we say a slide rule is a computer but an egg beater is not? —Churchland and Sejnowski (1992, p. 61)

… everyone who taps at a keyboard, opening a spreadsheet or a word-processing program, is working on an incarnation of a Turing machine …

—Time magazine, 29 March 1999, cited in Davis (2006a, p. 125)

✉ William J. Rapaport
rapaport@buffalo.edu
http://www.cse.buffalo.edu/∼rapaport/

[1] Department of Computer Science and Engineering, Center for Cognitive Science, University at Buffalo, The State University of New York, Buffalo, NY 14260-2500, USA

[2] Department of Philosophy, Center for Cognitive Science, University at Buffalo, The State University of New York, Buffalo, NY 14260-2500, USA

[3] Department of Linguistics, Center for Cognitive Science, University at Buffalo, The State University of New York, Buffalo, NY 14260-2500, USA

1 Introduction

A *Family Circus* comic strip shows a little boy holding a pencil: "Daddy said when he was a kid THIS was his computer. Is he joking?"[1] And an *Agnes* comic strip has the following dialogue between Agnes and her cynical friend Trout[2]:

> Agnes: I have a new computer!
> Trout: That's an old typewriter.
> Agnes: No … I'm fairly sure it is a computer.
> Trout: Must be wireless.
> Agnes: Apparently. See? I am preparing my first email on this wafer-thin screen.
> Trout: That's paper.

My philosophy of computer science course and textbook (Rapaport 2005b, 2018a)[3] examine some central philosophical questions about the nature of computing, computers, and computer science. They are designed to bring students and readers "up to speed" on a conversation about these issues, providing background information to the literature, so that they can become part of the conversation.

The present essay (adapted from Rapaport 2018a, Ch. 9) assumes that the reader knows what computing is and knows a bit of the history of computers, and it surveys some answers to the following question: If computer science is the study of computers, what *is* a computer? Are Churchland and Sejnowski's egg beaters, *Family Circus*'s pencils, or *Agnes*'s typewriters computers? Or are only *Time*'s "incarnations of Turing's machines" computers? Or is a computer something else altogether?

A fairly obvious, trivial, and almost-circular definition of 'computer' says that a computer is a machine that computes. The natural next question is: What does it mean to compute? But this shifts the burden of answering our question away from what *computers* are to the topic of what *computation* is. Many of the objections to various theories about *computers* are really objections to what counts as a *computation*. That question is, of course, interesting and important, but would take us too far afield (although I will say something about it at the end).[4] Thus, this survey is more concerned with the question of what a computer is, given a fixed meaning of 'compute'.

According to computer pioneer Arthur L. Samuel, in a 1953 article introducing computers to engineers who might have been unfamiliar with them,

> a computer … can be looked at from two different angles, which Professor Hartree has called the "anatomical" and the "physiological," that is, "of what is it made?" and "how does it tick?" (Samuel 1953, p. 1223, citing Hartree 1949, p. 56)

[1] http://familycircus.com/comics/march-6-2012/.

[2] http://www.gocomics.com/agnes/2013/3/7.

[3] See brief overviews of both in Rapaport (2017a, §1.2), and Rapaport (2017c, §1).

[4] I discuss it further in Rapaport (2018a, Chs. 7, 10, and 11), which examine the nature of algorithms and the Church–Turing Computability Thesis.

Samuel then goes on to describe the anatomy in terms of things like magnetic cores and vacuum tubes. Clearly, the anatomy has changed since 1953, so defining 'computer' "anatomically" in such terms doesn't seem to be the right way to go: It's too changeable. What's needed is a "physiological"—or functional—definition. At the very least, we might say that a computer is a physical machine (where, perhaps, it doesn't matter what it is made of) that is designed (i.e., engineered) to compute (i.e., to do computations) and, perhaps, that interacts with the world. (Rapaport 2017a explains why I add "perhaps" to this interaction clause.)

But does a computer *have* to be a "machine"? Does it *have* to be "engineered"? If the brain is a computer, then it would seem that computers could be *biological* entities (which, arguably, are not machines) that *evolved* (which, arguably, means that they were not engineered).[5] So, we should also ask whether the brain is a computer. But is it even correct to limit a computer to a *physical* device? Aren't Turing machines (TMs) computers? Or should we distinguish a "real" computer from a mathematical abstraction such as a TM? But, arguably, my iMac—which is surely a computer if anything is—*isn't* a TM. Rather, it can be *modeled* by a (universal) TM. And, to the extent that abstract TMs don't interact with the world, so much the worse for them as a model of what a computer is.[6] (Rapaport 2018a, §11.4.2 looks at the "hypercomputability" of interactive computation.)

But what about a *virtual* computer implemented in some software, such as a program that "simulates" a computer of a certain type (perhaps even a TM) but that runs on a (physical) computer of a very different type? For example, I once had my students use a "P88 Assembly Language Simulator"—a virtual machine whose programming language was "P88 Assembly Language" (Biermann 1990)—which was actually implemented in another virtual machine whose programming language was Pascal and which was, in turn, implemented on a physical Macintosh computer (Rapaport 2005a). Note that, ultimately, there is a physical substrate in these cases.

If the purpose of a computer is to compute, what kind of computations do they perform? Are they restricted to mathematical computations? Even if that's so, how much of a restriction is that? What I have called the binary-representation insight suggests that any (computable) information can be represented as a binary numeral (Rapaport 2017c, p. 14); hence, any computation on such information could be considered to be a mathematical computation.

And what about the difference between a "hardwired" TM that can only compute one thing and a "programmable" universal Turing machine (UTM) that can compute anything that is computable? Or what about the difference between a real, physical computer that can only compute whatever is *practically* computable (i.e., subject to reasonable space and time constraints) and an abstract, UTM that is not thus constrained?

And what about those egg beaters, or rocks? Surely, they are *not* computers. Or are they? In short, what is a computer?

[5] At least, not engineered by *humans*. Dennett (2017) would say that they *were* engineered—by Mother Nature, via the natural-selection algorithm.

[6] Thanks to my colleague Stuart C. Shapiro for many of these points.

2 Informal Definitions

2.1 Reference-Book Definitions

If you ask a random person what a computer is, they might try to describe their laptop.[7] The *Encyclopedia of Computer Science* says:

> A *digital computer* is a machine that will accept data and information presented to it in a discrete form, carry out arithmetic and logical operations on this data, and then supply the required results in an acceptable form. (Morris and Reilly 2000, p. 539)

And the *Oxford English Dictionary* offers these definitions[8]:

> **computer,** *n.*
>
> 1. A person who makes calculations or computations; a calculator, a reckoner; *spec.* a person employed to make calculations in an observatory, in surveying, etc. Now chiefly *hist*. [earliest citation dated 1613]
> 2. A device or machine for performing or facilitating calculation. [earliest citation dated 1869]
> 3. An electronic device (or system of devices) which is used to store, manipulate, and communicate information, perform complex calculations, or control or regulate other devices or machines, and is capable of receiving information (data) and of processing it in accordance with variable procedural instructions (programs or software); *esp.* a small, self-contained one for individual use in the home or workplace, used esp. for handling text, images, music, and video, accessing and using the Internet, communicating with other people (e.g. by means of email), and playing games. [earliest citation dated 1945]

We'll come back to these in Sect. 2.5.

2.2 Von Neumann's Definition

In his "First Draft Report on the EDVAC" (which—along with Turing 1936—may be taken as one of the founding documents of computer science), John von Neumann gave the following definition:

> An *automatic computing system* is a (usually highly composite) device, which can carry out instructions to perform calculations of a considerable order of complexity …. The instructions … must be given to the device in absolutely exhaustive detail. They include all numerical information which is required to solve the problem under consideration …. All these procedures require the use of some code to express … the problem …, as well as the necessary numerical material …. [T]he device … must be able to carry them out completely and

[7] Unless they don't realize that a laptop or a tablet *is* a computer! See the iPad advertisement at https://www.youtube.com/watch?v=sQB2NjhJHvY.

[8] http://www.oed.com/view/Entry/37975 (my bracketed interpolations).

> without any need for further intelligent human intervention. At the end of the required operations the device must record the results again in one of the forms referred to above. (von Neumann 1945, §1.0, p. 1)

Other comments (in this section of von Neumann 1945, as well as later, in §5.0, pp. 6ff) indicate that the code should be binary, hence that the computer is a "digital" device (§1.0, p. 1). This definition adheres closely to being a physical implementation of a TM, with clear allusions to the required algorithmic nature of the instructions, and with a requirement that there be both input and output. (Rapaport 2018a, §7.4 discusses the necessity—or lack thereof!—of this input-output (I/O) requirement.)

2.3 Samuel's Definition

Samuel's "physiological"—or functional—definition of a computer is this:

> an information or data processing device which accepts data in one form and delivers it in an altered form. (Samuel 1953, p. 1223)

This seems to be a very high-level description—perhaps *too* high a level: It omits any mention of computation or of algorithms. It does mention that the "delivered" data must have been "processed" from the "accepted" data by the "device"; so a computer is not just a mathematical *function* that relates the two forms of data—it's more of a function *machine*.[9] But there's no specification of the *kind* of processing that it does.

Partly because of this, and on purpose, it also doesn't distinguish between analog and digital computers. Samuel resolves this by adding the modifier 'digital', commenting that "Any operation which can be reduced to arithmetic or to simple logic can be handled by such a machine. There does not seem to be any theoretical limit to the types of problems which can be handled in this way" (Samuel 1953, p. 1224)—a nod, perhaps, to the binary-representation insight. Still, this doesn't limit the processing to *algorithmic* processing. It does, however, allow the *brain* to be considered as a computer: "when the human operator performs a reasonably complicated numerical calculation he [sic][10] is forcing his brain to act as a digital computer" (Samuel 1953, p. 1224).[11]

[9] The high-school characterization of a mathematical function as a "function machine" with a crank can be traced to Gödel (see Rapaport 2018a, §7.3.1.3 for discussion):

> [Turing] has shown that the computable functions defined in this way [i.e., in terms of TMs] are exactly those for which you can construct a machine with a finite number of parts which will do the following thing. If you write down any number $n_1, \ldots, n_r$ on a slip of paper and put the slip into the machine and turn the crank, then after a finite number of turns the machine will stop and the value of the function for the argument $n_1, \ldots, n_r$ will be printed on the paper. (Gödel 1938, p. 168; my bracketed interpolation)

[10] The use of the male gender here is balanced by Samuel's earlier statement that computers have "advantages in terms of the reductions in clerical manpower *and woman power*" (Samuel 1953, p. 1223; my italics).

[11] Cf. Chalmers (2011), quoted at the end of Sect. 5.1, below.

A bit later (p. 1225), he does say that the processing must be governed by rules; this gets closer to the notion of an algorithm, though he (so far) puts no constraints on the rules. It is only after he discusses the control unit of the computer and its programming (pp. 1226ff) that he talks about the kinds of control structures (loops, etc.) that are involved with algorithms. So, perhaps we could put all of this together and say that, for Samuel, a (digital) computer is a physical device that algorithmically processes digital data.

Further on, he adds the need for I/O devices (p. 1226). Are these really needed? Are they part of the abstract, mathematical model of a computer, namely, a TM? Your first reaction might be to say that the tape of a TM serves as a combined I/O device. But the tape is an integral part of the TM; it is really more like the set of internal switches of a physical computer than like an (external) I/O device. Physical computers normally have I/O devices as separate, additional components: Think of a computer like the Mac Mini, which is sold without a keyboard or a monitor.

But a computer with no I/O devices can only do batch processing of pre-stored data (if that—the Mac Mini can't do anything if there's no way to tell it to start doing something). Computers that interact with the external world require I/O devices, and that raises the question of their relationship to TMs. Briefly, interacting computers that halt or that have only computable input are simulable by TMs; interacting computers with *non*-computable input are equivalent to Turing's oracle machines (Turing 1939, pp. 172–173; Davis 2004).[12]

2.4 Davis's Characterization

Martin Davis suggests (but does not explicitly endorse) the idea that computers are simply any devices that "carry out algorithms" (Davis 2000, pp. 366–367). Of course, this depends on what 'carries out' means: Surely it has to include as part of its meaning that the internal mechanism of the device must operate in accordance with—must behave exactly like—one of the logically equivalent mathematical models of computation. Surely, any computer does that. But is anything that does that a computer? Can a computer be defined (merely) as a set of registers with contents or switches with settings? If they are binary switches, each is either on or else off; computation changes the contents (the settings). (See Sect. 4.2, below.) Do some of the register contents or switch settings have to be interpreted as data, some as program, and the rest as irrelevant (and some as output?). Who (or what) does the interpreting?

2.5 Discussion

One common thread in informal definitions such as these is that computers are:

1. devices or machines …
2. … that take input (data, information),
3. process it (manipulate it; or operate, calculate, or compute with it) …

[12] For further discussion, see Rapaport (2018a, Ch. 11).

4. … in accordance with instructions (a program),
5. and then output a result (presumably, more data or information, but also including control of another device).

There are some other features that are usually associated with "computers": The kind that we are interested in must be, or typically are:

automatic There is no human intervention (beyond, perhaps, writing the program). Of course, the holy grail of programming is to have self-programmed computers, possibly to include having the "desire" or "intention" to program themselves (as in science fiction). Humans might also supply the input or read the output, but that hardly qualifies as "intervention". (Rapaport 2018a, Ch. 11, explores "intervention" in the guise of "interactive" computing.)

general purpose A computer must be capable of *any* processing that is "algorithmic", by means of a suitable program. This is the heart of Turing's universal machine. Recall that a TM "runs" only *one* program. The UTM is also a TM, so it, too, also runs only one program, namely, the fetch-execute cycle that enables the *simulation* of *another* (i.e., *any* other) single-program TM.

physically efficient Many lists of computer features such as this one say that computers are *electronic.* But that is a matter of "anatomy". Modern computers are, as a matter of fact, electronic, but there is work on quantum computers, optical computers, DNA computers, etc. So, being electronic is not essential. The crucial ("physiological") property is, rather, to be constructed in such a way as to allow for high processing speeds or other kinds of physical efficiencies.

digital Computers should process information expressed in discrete, symbolic form (typically, alpha-numeric form, but perhaps also including graphical form). The contrast is typically with being "analog", where information is represented by means of continuous physical quantities.

algorithmic What about the "calculations", the "arithmetic and logical operations"? Presumably, these need to be algorithmic, though neither the *OED* nor the *Encyclopedia of Computer Science* definitions say so. And it would seem that the authors of those definitions have in mind calculations or operations such as addition, subtraction, etc.; maybe solving differential equations; Boolean operations involving conjunction, disjunction, etc; and so on. These require the data to be numeric (for math) or propositional or truth-functional (for Boolean and logical operations), at least in some "ultimate" sense: I.e., any *other* data (pictorial, etc.) must be encoded numerically or propositionally, or else would need to allow for other kinds of operations.

Do computers *have to* be algorithmic? There are three things to consider in this regard. The first concerns "heuristics", often thought of as "rules of thumb"[13] that can "leapfrog the step-by-step programming that characterizes computers" (as one referee put it). The objection is that anything that relies on such heuristics (e.g., humans!) are not, or cannot be, computers. But it doesn't follow that computers can't use heuristics, albeit perhaps in a slightly different sense:

> [S]ome functions might only be practically computable "indirectly" via a "heuristic": A *heuristic for problem p* can be defined as an *algorithm* for some problem p', where the solution to p' is "good enough" as a solution to p (Rapaport 1998, p. 406). Being "good enough" is, of course, a subjective notion; Oommen and Rueda (2005, p. 1) call the "good enough" solution "a *sub-optimal* solution that, hopefully, is arbitrarily close to the *optimal*." The idea is related to Herbert Simon's notion of bounded rationality (Simon 1996): We might not be able to solve a problem p because of limitations in space, time, or knowledge, but we might be able to solve a *different* problem p' algorithmically within the required spatio-temporal-epistemic limits. And if the *algorithmic* solution to p' gets us closer to a solution to p, then it is a *heuristic* solution to p. But it is still an algorithm. (For more on heuristics, see Romanycia and Pelletier 1985; Chow 2015.) —Rapaport (2017c, §14.1.3)

The second consideration against algorithmicity concerns "super-Turing" computation or "hypercomputation" (Copeland 2002), understood as the ability of Turing *machinery* (Hintikka and Mutanen 1997, p. 175; Kugel 2002, p. 566) to perform "computation-like processes" (Davis 2006b, p. 4), perhaps via relativistic "computers" (Hogarth 1992), "interactive" computers (Wegner 1997), trial-and-error computers (Putnam 1965; Hintikka and Mutanen 1997; Kugel 2002), or oracle machines. But this topic would take us too far afield. (For further discussion, see Rapaport 2018a, Ch. 11.)

The third consideration (related to the first two) is whether Turing computation is sufficient for creative endeavors, including the ability to do creative mathematics. Turing's comments on this topic are a bit enigmatic:

> [O]ne can show that however the machine [i.e., a computer] is constructed there are bound to be cases where the machine fails to give an answer [to a mathematical question], but a mathematician would be able to. On the other hand, the machine has certain advantages over the mathematician. Whatever it does can be relied upon, assuming no mechanical 'breakdown', whereas the mathematician makes a certain proportion of mistakes. I believe that this danger of the mathematician making mistakes is an unavoidable corollary of his [sic] power of sometimes hitting upon an entirely new method. (Turing 1951, p. 256)

[13] A curious phrase: In contemporary American English, it refers to hints, suggestions, informal rules (often with many exceptions), as in Polya (1957). But Turing (1947, p. 383) used the phrase to refer to algorithms!

In any case, many of these issues are more concerned with the nature of *computation*, rather than the nature of *computers*.[14]

There are clear cases of things that *are* computers, both *digital* and *analog*. For example, Macs, PCs, etc. are clear cases of digital computers. And slide rules and certain machines at various universities are clear cases of analog computers. (However, such analog computers may be mostly of historical interest and don't seem to be programmable—i.e., universal, in Turing's sense.)[15]

And there seem to be clear cases of things that are *not* computers: I would guess that most people would not consider rocks, walls, ice cubes, egg beaters, or solid blocks of plastic to be computers. (Note that I said "most" people!)

And there are even clear cases of devices for which it might be said that it is not clear whether, or in what sense, they are computers, such as Atanasoff and Berry's ABC (Wheeler 1997).

So: What is a computer? What is the relation of a computer to a TM and to a UTM? Is the (human) brain a computer? Is your smartphone a computer? *Could* a rock or a wall be considered to be a computer? Might *anything* be a computer? Might *everything*—such as the universe itself—be a computer? Or are some of these just badly formed questions?[16]

3 Computers, TMs, and UTMs

> All modern general-purpose digital computers are physical embodiments of the same logical abstraction[:] Turing's universal machine. (Robinson 1994, pp. 4–5)

3.1 Computers as TMs

Let us try our hand at a more formal definition of 'computer'. An obvious candidate for such a definition is this:

[14] See Rapaport (2018a, Ch. 11) for further discussion.

[15] Contrary to what one referee suggested, not being programmable doesn't rule out brains by definition as computers; surely(?), brains are programmable! (See Sect. 2.3, above, and Sect. 5.1, below. On the "surely" operator, see Dennett 2013, Ch. 10.)

On analog computers, see Rubinoff (1953), Samuel (1953, p. 1224, § "The Analogue Machine"), Jackson (1960), Pour-El (1974), Copeland (1997, "Nonclassical Analog Computing Machines", pp. 699–704), Hedger (1998), Shagrir (1999), Holst (2000), Stoll (2006), Care (2007), Piccinini (2008), Fortnow (2010), Piccinini (2011), McMillan (2013), Corry (2017); and http://hrl.harvard.edu/analog/. For alternative ways to compute with real numbers other than with analog computers, see Blum et al. (1989), Buzen (2011).

[16] For other attempts at defining 'computer', see Shagrir (1999), Harnish (2002), Anderson (2006), Kanat-Alexander (2008), Chalmers (2011, "What about computers?", pp. 335–336, 2012), Egan (2012), Rescorla (2012), Scheutz (2012), Shagrir (2012) and Chirimuuta et al. (2014).

(DC0) A computer is *any physical device that computes.*

Because a TM is a mathematical model of what it means to compute, we can make this a bit more precise:

(DC1) A computer is *an implementation of a TM.*

A TM is an abstract, mathematical structure. An "implementation" of an *abstract* object is (usually) a *physical* object that satisfies the definition of the abstract one. (The hedge-word 'usually' is there in order to allow for the possibility of non-physical—or "virtual"—software implementations of a TM.)[17] So, a physical object that satisfies the definition of a TM would be an "implementation" of one.

Of course, no physical object can fully satisfy that definition if part of the definition requires it to be "perfect" in the following sense:

> A Turing machine is like an actual digital computing machine, except that (1) it is error free (i.e., it always does what its table says it should do), and (2) by its access to an unlimited tape it is unhampered by any bound on the quantity of its storage of information or "memory". (Kleene 1995, p. 27)

The type-(2) limitation of a "real" (physical) TM is not a very serious one, given (a) the option of always buying another square to staple to the tape and (b) the fact that no (halting) computation could require an actual infinity of squares (else it would not be a finite computation). A more significant type-(2) limitation is that some computations might require more squares than there could be in the universe (as is the case with an algorithm for playing perfect chess (Zobrist 2000, p. 367)). The type-(1) limitation of a "real" TM—being error free—does not obviate the need for program verification. Even an "ideal" TM could be poorly programmed. (On program verification, see Rapaport 2018a, Ch. 16.)

So let's modify our definition to take care of this:

(DC2) A computer is a "*physically plausible*" implementation of a TM

where 'physically plausible' is intended to summarize those physical limitations.

Let's now consider two questions:

- Is a TM a computer?
- Is a Mac (or a PC, or any other real computer) a physically plausible implementation of a TM?

The first question we can dismiss fairly quickly: TMs are not physical objects, so they can't be computers. A TM is, of course, a mathematical model of a computer. (But a virtual, software implementation of a TM *is*, arguably, a computer.)

[17] Rapaport (1999, 2005a) argue that implementation is semantic interpretation.

The second question is trickier. Strictly speaking, the answer is 'no', because Macs (and PCs, etc.) don't behave the way that TMs do. They actually behave more like another mathematical model of computation: a register machine (Wang 1957; Shepherdson and Sturgis 1963). Register machines, however, are logically equivalent to TMs; they are just another mathematical model of computation.[18] Moreover, other logically equivalent models of computation are even further removed from TMs or register machines: How might a computer based on recursive functions work? Or one based on the lambda calculus? (Think of Lisp Machines.) This suggests a further refinement to our definition:

(DC3) A computer is a physically plausible implementation of *anything logically equivalent to* a TM.

There is another problem, however: Computers, in any informal sense of the term (think laptop or even mainframe computer) are programmable. TMs are not!

But *universal* TMs are! The ability to store a program on a universal TM's tape makes it programmable; i.e., the UTM can be changed from simulating the behavior of one TM to simulating the behavior of a different one. A computer in the modern sense of the term really means a *programmable* computer, so here is a slightly better definition:

(DC4) A (programmable) computer is a physically plausible implementation of anything logically equivalent to a *universal* TM.

But a program need not be stored physically *in* the computer: It could "control" the computer via a wireless connection from a different location. The ability to store a program *in* the computer *along with* the data allows for the program to change *itself*. Moreover, a hardwired, non-universal computer could be programmed by re-wiring it, assuming that the wires are manipulable (Moor 1978). That's how early mainframe computers (like ENIAC) were programmed. So, this raises another question (not discussed here): What exactly is a "stored-program" computer, and does it differ from a "programmable" computer?[19]

One consideration not yet taken into account here is that, to be of practical use, a computer should be capable of interaction with the external world. But this is not a requirement. Surely(?), a non-interactive device that satisfies (DC4) would be a computer, albeit a special-purpose, batch-processing one.

In the next section, we look at three recent attempts in the philosophical literature to define 'computer'. In Sect. 5, we will briefly consider two non-standard, alleged examples of computers: brains and the universe itself.

[18] Register machines—although better models of real computers than TMs are—do not, as one referee implied, have fixed-size registers (as real computers do); they are mathematical idealizations.

[19] For discussion of this, see von Neumann (1945, §2.3, p. 2), Carpenter and Doran (1977, p. 270), Randell (1994), Copeland (2013), Daylight (2013), Haigh (2013), Vardi (2013), and Rapaport (2018a, §9.4.2).

4 Three Recent Philosophical Definitions

4.1 Searle: Anything is a Computer

4.1.1 Searle's Argument

John Searle's "Is the Brain a Digital Computer?" (1990), covers a lot of ground and makes a lot of points about the nature of computers, the nature of the brain, the nature of cognition, and the relationships among them. In this section, we are going to focus on what Searle says about the nature of computers, with only a few side glances at the other issues.[20]

Here is Searle's argument relevant to our main question about what a computer is:

1. Computers are described in terms of 0s and 1s. (See Searle 1990, p. 26.)

   Taken literally, he is saying that computers are described in terms of certain *numbers*. Perhaps he should have said that computers are described in terms of '0's and '1's. In other words, perhaps he should have said that computers are described in terms of certain *numerals*.
2. Therefore, being a computer is a syntactic property. (See Searle 1990, p. 26.)

   Syntax is the study of the properties of, and relations among, symbols or uninterpreted marks on paper (or in some other medium); a rough synonym is 'symbol manipulation' (Rapaport 2017b). In line with the distinction between numbers and numerals, note that only numerals are symbols.
3. Therefore, being a computer is not an "intrinsic" property of physical objects. (See Searle 1990, pp. 27–28.)
4. Therefore, *we can ascribe* the property of being a computer *to* any object. (See Searle 1990, p. 26.)
5. Therefore, everything is a computer. (See Searle 1990, p. 26.)

Of course, this doesn't quite answer our question, "What *is* a computer?". Rather, the interpretation and truth value of these theses will depend on what Searle thinks a computer is. Let's look at exactly what Searle says about these claims.

4.1.2 Computers are Described in Terms of 0s and 1s

After briefly describing TMs as devices that can perform the actions of printing '0' or '1' on a tape and of moving left or right on the tape, depending on conditions specified in its program, Searle says this:

> If you open up your home computer you are most unlikely to find any 0's and 1's or even a tape. But this does not really matter for the definition. To find out if an object is really a digital computer, it turns out that we do not actually

[20] For more detailed critiques and other relevant commentary, see Piccinini (2006, 2007b, 2010), and Rapaport (2007).

> have to look for 0's and 1's, etc.; rather we just have to look for something that **we** could *treat as* or *count as* or *could be used to* function as 0's and 1's. (Searle 1990, p. 25; my boldface, Searle's italics)

So, according to Searle, a computer is a physical object that can be *described* as a TM.

As Hilbert famously observed about geometry, "One must be able to say at all times—instead of points, lines, and planes—tables, chairs, and beer mugs".[21] So, too, in the case of TMs, one must be able to say at all times—instead of tapes, squares, and symbols—tables, chairs, and beer mugs. But I'll use place settings instead of chairs; it will make more sense, as you will see. In this tavern metaphor, a TM, we might say, must have a table.[22] Each table must have a sequence of place settings associated with it (so we must be able to talk about the *n*th place setting at a table). And each place setting can have a beer mug on it; there might be different kinds of beer mugs, but they have to be able to be distinguished from each other, so that we don't confuse them. In other words, *it is the logical or mathematical* ***structure*** *of a computing machine that matters, not what it is made of.* (As Samuel and Hartree would say, it's the physiology that matters, not the anatomy.) So, a "tape" doesn't have to be made of paper (it could be a wooden table), a "square" doesn't have to be a regular quadrilateral that is physically part of the "tape" (it could be a place setting at the table), and "symbols" only have to be such that a "square" can "bear" one (e.g., a numeral can be written on a square of the tape, or a beer mug can be placed at a place setting belonging to the table).

Thus, anything that satisfies the definition of a TM *is* a TM, whether it has a paper tape divided into squares with the symbols '0' or '1' printed on them or whether it is a table and placemats with beer mugs on them (or whether it consists of toilet paper and pebbles (Weizenbaum 1976, Ch. 2)). All we need is to be able to "treat" some part of the physical object as *playing the role of* the TM's '0's and '1's. So far, so good.

Or is it? Is your home computer *really* a TM? Or is it a device whose behavior is "merely" *logically equivalent* to that of a TM? I.e., is it a device that can compute all and only the functions that a TM can compute, even if it does so differently from the way that a TM does? There are many different mathematical models of computation: TMs, register machines, the lambda calculus, and recursive functions are just a few of them. Suppose someone builds a computer that operates in terms of recursive functions instead of in terms of a TM. I.e., it can compute successors, predecessors, and projection functions, and it can combine these using generalized composition, conditional definition, and while-recursion, instead of printing '0's and '1's, moving left and right, and combining these using "go to" instructions (changing from one *m*-configuration to another (Turing 1936)). (See Rapaport 2018a, Ch. 7.) Both the recursive-function computer, as well as your home computer (with a "von Neumann" architecture, whose method of computation uses the primitive machine-language instructions and control structures of, say, an Intel

[21] Cf. Hilbert's *Gesammelte Abhandlungen*, vol. 3, p. 403, as cited in Coffa (1991, p. 135); cf. Stewart Shapiro (2009, p. 176).

[22] Not to be confused with the TM's "*machine* table", i.e., its hardwired program.

chip), are logically equivalent to a TM, in the sense of having the same I/O behavior, but their "internal" behaviors are radically different. We can ask: Are recursive-function computers, TMs, Macs, and PCs not only *extensionally* equivalent but also *intensionally* equivalent? Can we really describe the recursive-function computer and your home computer in terms of a TM's '0's and '1's? Or are we limited to showing that anything that the recursive-function computer and your home computer can compute can also be computed by a TM (and vice versa)—but not necessarily *in the same way*?

So, something might be a computer without being "described in terms of '0's and '1's", depending on exactly what you mean by 'described in terms of'. Perhaps Searle should have said something like this: Computers are described in terms of the primitive elements of the mathematical model of computation that they implement. But let's grant him the benefit of the doubt and continue looking at his argument.

4.1.3 Being a Computer is a Syntactic Property

Let us suppose, for the sake of the argument, that computers are described in terms of '0's and '1's. Such a description is *syntactic*. This term (which pertains to symbols, words, grammar, etc.) is usually contrasted with 'semantic' (which pertains to meaning), and Searle emphasizes that contrast early in his essay when he says that "syntax is not the same as, nor is it by itself sufficient for, semantics" (Searle 1990, p. 21).[23]

But Searle uses the term 'syntactic' as a contrast to being *physical*. Just as there are many ways to be computable (TMs, recursive functions, lambda calculus, etc.)—all of which are equivalent—so there are many ways to be a carburetor. "A carburetor … is a device that blends air and fuel for an internal combustion engine" (http://en.wikipedia.org/wiki/Carburetor), *but it doesn't matter what it is made of*, as long as it can perform that blending "function" (purpose). "[C]arburetors can be made of brass or steel" (Searle 1990, p. 26); they are "multiply realizable"—that is, you can "realize" (or make) one in "multiple" (or different) ways. They "are defined in terms of the production of certain *physical* effects" (Searle 1990, p. 26).

> But the class of computers is defined **syntactically** in terms of the *assignment* of 0's and 1's. (Searle 1990, p. 26; Searle's italics, my boldface)

In other words, if something is defined in terms of symbols, like '0's and '1's, then it is defined in terms of syntax, not in terms of what it is physically made of.

Hence, being a computer is a syntactic property, not a physical property. It is a property that something has in virtue of … of what? There are two possibilities, given what Searle has said. First, perhaps being a computer is a property that something has in virtue of *what it **does***: its *function* or *purpose* (or "physiology"). Second, perhaps being a computer is a property that something has in virtue of *what someone **says** that it does*: how it is *described*. But what something *actually* does may be different from what someone *says* that it does.

[23] However, for arguments that syntax *does* "suffice" for semantics—that semantics *is* a kind of syntax—see Rapaport (1988, 2017b).

So, does Searle think that something is a computer in virtue of its *function* or in virtue of its *syntax*? Suppose you find a black box with a keyboard and a screen in the desert (cf. Weizenbaum 1976, Ch. 5) and that, by experimenting with it, you decide that it displays on its screen the greatest common divisor (GCD) of two numbers that you type into it. It certainly seems to *function* as a computer (as a TM for computing GCDs). And you can probably describe it in terms of '0's and '1's, so you can also *say* that it is a computer. It seems that *if something functions as a computer, then you can describe it in terms of '0's and '1's.*

What about the converse? If you can *describe* something in terms of '0's and '1's, does it *function* as a computer? Suppose that the black box's behavior is inscrutable: The symbols on the keys are unrecognizable, and the symbols displayed on the screen don't seem to be related in any obvious way to the input symbols. But suppose that someone manages to invent an interpretation of the symbols in terms of which the box's behavior can be described as computing GCDs. Is "computing GCDs" really what it does? Might it not have been created by some extraterrestrials solely for the purpose of entertaining their young with displays of pretty pictures (meaningless symbols), and that it is only by the most convoluted (and maybe not always successful) interpretation that it can be described as computing GCDs?

You might think that the box's *function* is more important for determining what it is. But Searle thinks that our ability to *describe* it syntactically is more important! After all, whether or not the box was *intended* by its creators to compute GCDs or to entertain toddlers, if it can be accurately *described* as computing GCDs, then, in fact, it computes GCDs (as well as, perhaps, entertaining toddlers with pretty pictures).[24]

Again, let's grant this point to Searle. He then goes on to warn us:

> But this has two consequences which might be disastrous:
>
> 1. The same principle that implies multiple realizability would seem to imply universal realizability. If computation is defined in terms of the assignment of syntax then everything would be a digital computer, because any object whatever could have syntactical ascriptions made to it. You could describe anything in terms of 0's and 1's.
> 2. Worse yet, syntax is not intrinsic to physics. The ascription of syntactical properties is always relative to an agent or observer who treats certain physical phenomena as syntactical. (Searle 1990, p. 26)

Let's take these in reverse order.

[24] An alternative view of this was given in Goodman (1987, p. 484):

> Suppose that a student is successfully doing an exercise in a recursive function theory course which consists in implementing a certain Turing machine program. There is then no reductionism involved in saying that he is carrying out a Turing machine program. He intends to be carrying out a Turing machine program. … Now suppose that, unbeknownst to the student, the Turing machine program he is carrying out is an implementation of the Euclidean algorithm. His instructor, looking at the pages of more or less meaningless computations handed in by the student, can tell from them that the greatest common divisor of 24 and 56 is 8. The student, not knowing the purpose of the machine instructions he is carrying out, cannot draw the same conclusion from his own work. I suggest that the instructor, but not the student, should be described as carrying out the Euclidean algorithm. (This is a version … of Searle's Chinese room argument ….)

4.1.4 Being a Computer is Not an Intrinsic Property of Physical Objects

According to Searle, being a computer is not an intrinsic property of physical objects, because being a computer is a syntactic property, and "syntax is not intrinsic to physics". What does that quoted thesis mean, and why does Searle think that it is true?

What is an "intrinsic" property? Searle doesn't tell us, though he gives some examples:

> [G]reen leaves intrinsically perform photosynthesis[;] … hearts intrinsically pump blood. It is not a matter of us arbitrarily or "conventionally" assigning the word "pump" to hearts or "photosynthesis" to leaves. There is an actual fact of the matter. (Searle 1990, p. 26)

So, perhaps "intrinsic" properties are properties that something "really" has as opposed to merely being *said* to have, much the way our black box in the previous section may or may not "really" compute GCDs but can be *said* to compute them. But what does it mean to "really" have a property?

Are Searle's "intrinsic" properties *essential* ones? Perhaps he is saying that being a computer is not an essential property of an object, but only an accidental property.

Could his "intrinsic" properties be something like Locke's *primary* qualities (as opposed to secondary or relational properties)? Perhaps Searle is saying that being a computer is only a relational property of an object.

Is being a computer a *natural kind*? There probably aren't any computers in nature (unless the brain is a computer; and see Sect. 5.2 on whether nature itself is a computer), but there may also not be any prime numbers in nature, yet mathematical objects are something thought to exist independently of human thought. If they do, then, because a TM is a mathematical object, it might exist independently of human thought and, hence, being a computer might be able to be considered to be a "natural" mathematical kind. Perhaps Searle is saying that being a computer is not a natural kind in one of these senses.[25]

In fact, a computer is probably not a natural kind for a different reason: It is an artifact, something created by humans. But the nature of artifacts is controversial: Clearly, chairs, tables, skyscrapers, atomic bombs, and pencils are artifacts; you don't find them in nature, and if humans had never evolved, there probably wouldn't be any of these artifacts. But what about bird's nests, beehives, beaver dams, and other such things constructed by non-human animals? What about "socially constructed" objects like money? One of the crucial features of artifacts is that what they *are* is relative to what a person *says* they are. You won't find a table occurring naturally in a forest, but if you find a tree stump, you might use it as a table. So, something might be a computer, Searle might say, only if a human uses it that way or can describe it as one. In fact, Searle does say this:

[25] For more discussion on what 'intrinsic' means, see Lewis (1983), Langton and Lewis (1998), Skow (2007), Bader (2013), Marshall (2016), Weatherson and Marshall (2018).

> [W]e might discover in nature objects which had the same sort of shape as chairs and which could therefore be used as chairs; but we could not discover objects in nature which were functioning as chairs, except relative to some agents who regarded them or used them as chairs. (Searle 1990, p. 28)

Why does Searle think that syntax is not "intrinsic" to physics? Because " 'syntax' is not the name of a physical feature, like mass or gravity. … [S]yntax is essentially an observer relative notion" (Searle 1990, p. 27). I think that what Searle is saying here is that we can analyze physical objects in different ways, no one of which is "privileged" or "more correct"; i.e., we can carve nature into different joints, in different ways. On some such carvings, we may count an object as a computer; on others, we wouldn't. By contrast, an object has mass independently of how it is described: *Having* mass is *not* relative to an observer. *How* its mass is measured *is* relative to an observer.

But couldn't being a computer be something like that? There may be lots of different ways to measure mass, but an object always has a certain quantity of mass, no matter whether you measure it in grams or in some other units. In the same way, there may be lots of different ways to measure length, but an object always has a certain length, whether you measure it in centimeters or in inches. Similarly, an object (natural or artifactual) will have a certain structure, whether you describe it as a computer or as something else. If that structure satisfies the definition of a TM, then it *is* a TM, no matter how anyone *describes* it.

Searle anticipates this reply:

> [S]omeone might claim that the notions of "syntax" and "symbols" are just a manner of speaking and that what we are really interested in is the existence of systems with discrete physical phenomena and state transitions between them. On this view we don't really need 0's and 1's; they are just a convenient shorthand. (Searle 1990, p. 27)

Compare this to my example above: Someone might claim that specific units of measurement are just a manner of speaking and that what we are really interested in is the actual length of an object; on this view, we don't really need centimeters or inches; they are just a convenient shorthand.

Searle replies:

> But, I believe, this move is no help. **A physical state of a system is a computational state only relative to the assignment to that state of some computational role, function, or interpretation.** The same problem arises without 0's and 1's because notions such as computation, algorithm and program do not name intrinsic physical features of systems. Computational states are not *discovered within* the physics, they are *assigned* to the physics. (Searle 1990, p. 27, my boldface, Searle's italics.)

But this just repeats his earlier claim; it gives no new reason to believe it. He continues to insist that being a computer is more like "inches" than like length.

So, we must ask again: Why does Searle think that syntax is not intrinsic to physics? Perhaps, if a property is intrinsic to some object, then that object can only

have the property in one way. For instance, color is presumably not intrinsic to an object, because an object might have different colors depending on the conditions under which it is perceived. But the physical structure of an object that causes it to reflect a certain wavelength of light is always the same; that physical structure is intrinsic. On this view, here is a reason why syntax might not be intrinsic: The syntax of an object is, roughly, its abstract structure (Rapaport 2017b). But an object might be able to be understood in terms of several different abstract structures (and this might be the case whether or not human observers assign those structures to the object). If an object has no unique syntactic structure, then syntax is not intrinsic to it. But if an object has (or can be assigned) a syntax of a certain kind, then it does have that syntax even if it also has another one. And if, under one of those syntaxes, the object is a computer, then it *is* a computer.

But that leads to Searle's next point.

4.1.5 We Can Ascribe the Property of Being a Computer to Any Object

There is some slippage in the move from "syntax is not intrinsic to physics" to "we can ascribe the property of being a computer to any object". Even if syntax is not intrinsic to the physical structure of an object (perhaps because a given object might have several different syntactic structures), why must it be the case that *any* object *can* be ascribed the syntax of being a computer? But is it really the case that *anything* can be described in terms of '0's and '1's?

One reason might be this: Every object has (or can be ascribed) *every* syntax. That seems to be a very strong claim. To refute it, however, all we would need to do is to find an object *O* and a syntax *S* such that *O* lacks (or cannot be ascribed) *S*. One possible place to look would be for an *O* whose "size" in some sense is smaller than the "size" of some *S*. I will leave this as an exercise for the reader: If you can find such *O* and *S*, then I think you can block Searle's argument at this point.

Here is another reason why *any* object might be able to be ascribed the syntax of being a computer: There might be something special about the syntax of being a computer—i.e., about the formal structure of TMs—that does allow it to be ascribed to (or found in) any object. This may be a bit more plausible than the previous reason. After all, TMs are fairly simple. Again, to refute it, we would need to find an object *O* such that *O* lacks (or cannot be ascribed) the syntax of a TM. Searle thinks that we cannot find such an object.

4.1.6 Everything is a Computer

> Unlike computers, ordinary rocks are not sold in computer stores and are usually not taken to perform computations. Why? What do computers have that rocks lack, such that computers compute and rocks don't? (If indeed they don't?) … A good account of computing mechanisms should entail that paradigmatic examples of computing mechanisms, such as digital computers, calculators, both universal and non-universal Turing machines, and finite state automata, compute. … A good account of computing mechanisms should entail that all paradigmatic examples of non-computing mechanisms and

> systems, such as planetary systems, hurricanes, and digestive systems, don't perform computations. (Piccinini 2015, pp. 7, 12)

We can ascribe the property of being a computer to any object if and only if everything is a computer.

> Thus for example the wall behind my back is right now implementing the Wordstar program, because there is some pattern of molecule movements which is isomorphic with the formal structure of Wordstar. (Searle 1990, p. 27)

Searle does not offer a detailed argument for how this might be the case, but other philosophers have done so (e.g., Putnam (1988, Appendix), Chalmers (1996, 2011), Suber (1997)). Let's assume, for the moment, that it can be done.

In that case, things are not good, because this trivializes the notion of being a computer. If everything has some property P, then P isn't a very interesting property; P doesn't help us categorize the world, so it doesn't help us understand the world:

> [A]n objection to Turing's analysis… is that although Turing's account may be necessary it is not sufficient. If it is taken to be sufficient then too many entities turn out to be computers. The objection carries an embarrassing implication for computational theories of mind: such theories are devoid of empirical content. If virtually anything meets the requirements for being a computational system then wherein lies the explanatory force of the claim that the brain is such a system? (Copeland 1996, §1, p. 335)

So, x is a computer iff x is a (physical) model of a TM. To say that this "account" is "necessary" means that, if x is a computer, then it is a model of a TM. That seems innocuous. To say that it is a "sufficient" account is to say that, if x is a model of a TM, then it is a computer. This is allegedly problematic, because, allegedly, anything can be gerrymandered to make it a model of a TM; hence, anything is a computer (including, for uninteresting reasons, the brain).

How might we respond to this situation? One way is to bite the bullet and accept that, under some description, any object (even the wall behind me) can be considered to be a computer. And not just some specific computer, such as a TM that executes the Wordstar program:

> [I]f the wall is implementing Wordstar then if it is a big enough wall it is implementing any program, including any program implemented in the brain. (Searle 1990, p. 27)

If a big enough wall implements any program, then it implements the UTM!

But perhaps this is OK. After all, there is a difference between an "intended" interpretation of something and what I will call a "gerrymandered" interpretation. For instance, the intended interpretation of Peano's axioms for the natural numbers is the sequence $\langle 0, 1, 2, 3, \ldots \rangle$. There are also many other "natural" interpretations, such as $\langle$I, II, III, …$\rangle$, or $\langle \varnothing, \{\varnothing\}, \{\{\varnothing\}\}, \ldots \rangle$, or $\langle \varnothing, \{\varnothing\}, \{\varnothing, \{\varnothing\}\}, \ldots \rangle$, and so on. But extremely contorted ones, such as a(n infinite) sequence of all numeral

names in French arranged alphabetically, are hardly "good" examples. Admittedly, they *are* examples of natural numbers, but not very useful ones. (For further discussion, see Benacerraf 1965.)

A better reply to Searle, however, is to say that he's wrong: Some things are *not* computers. Despite what he said in the last passage quoted above, the wall behind me is *not* a UTM; I really cannot use it to post to my Facebook account or to write a letter, much less to add $2 + 2$. It is an empirical question whether something actually behaves as a computer. And the same goes for other syntactic structures. Consider the formal definition of a mathematical group:

> A *group* is_{def} a set of objects (e.g., integers) that is closed under an associative binary operation (e.g., addition), that has an identity element (e.g., 0), and is such that every element of the set has an inverse (e.g., in the case of integer n, its inverse is $-n$).

Not every set is a group. Similarly, there is no reason to believe that everything is a TM.

> In order for the system to be used to compute the addition function these causal relations have to hold *at a certain level of grain*, a level that is determined by the discriminative abilities of the user. That is why … no money is to be made trying to sell a rock as a calculator. Even if (*per mirabile*) there happens to be a set of state-types at the quantum-mechanical level whose causal relations do mirror the formal structure of the addition function, microphysical changes at the quantum level are *not* discriminable by human users, hence human users could not use such a system to add. (God, in a playful mood, could use the rock to add.) (Egan 2012, p. 46)

Chalmers (2012, pp. 215–216) makes much the same point:

> On my account, a pool table will certainly implement various a-computations [i.e., computations as abstract objects] and perform various c-computations [i.e., concrete computational processes]. It will probably not implement interesting computations such as algorithms for vector addition, but it will at least implement a few multi-state automata and the like. These computations will not be of much explanatory use in understanding the activity of playing pool, in part because so much of interest in pool are not organizationally invariant and therefore involve more than computational structure.

In other words, even if Searle's wall implements Wordstar, *we* wouldn't be able to use it as such.

4.1.7 Ohter Views in the Vicinity of Searle's

> We count something as a computer because, and only when, its inputs and outputs can usefully and systematically be interpreted as representing the ordered pairs of some function that interests us. … This means that delimiting the class of computers is not a sheerly empirical matter, and hence that

> "computer" is not a natural kind. … Similarly, we suggest, there is no intrinsic property necessary and sufficient for all computers, just the interest-relative property that someone sees value in interpreting a system's states as representing states of some other system, and the properties of the system support such an interpretation. … [I]n this very wide sense, even a sieve or a threshing machine [or an egg beater!?] could be considered a computer…. (Churchland and Sejnowski 1992, pp. 65–66; my bracketed interpolation)

This is essentially Searle's point, only with a positive spin put on it. Note that the definition in the first sentence has an objective component: The inputs and outputs must be computationally related. Note, too, that no specification is placed on whether the mechanism by which the inputs are transformed into the outputs is a computational one. The definition also has a subjective component: If the function computed by the alleged computer is of no human interest, then it is not a computer!

Thus, this is a bit different from Searle: Where Searle says that the wall behind me is (or can be interpreted as) a word processor, Churchland and Sejnowski say that the wall behind me *is* computing something, but—to the extent that we don't care what it is—we don't bother considering it to be a computer.

Presumably, the wall behind me doesn't have to *be* a computer in order for its (molecular or subatomic) behavior to be *describable* computationally. Or is Searle making a stronger claim, namely, that, not only is its behavior *describable* computationally, but it *is* a computation? Dana Ballard (1997, p. 11) has an interesting variation on that stronger claim:

> Something as ordinary as a table might be thought of as running an algorithm that adjusts its atoms continually, governed by an energy function. Whatever its variables are, just denote them collectively by x. Then you can think of the table as solving the problem of adjusting its atoms so as to minimize energy, that is, $\min_x E(x)$. Is this computation?

Note that this is different from Searle's claim that the table (or a wall) might be computing a word processor. It seems closer to the idea that the solar system might be computing Kepler's law (see Sect. 5.2, below).

Another claim in the vicinity of Searle's and Ballard's concerns DNA computing:

> Computer. The word conjures up images of keyboards and monitors. … But must it be this way? The computer that you are using to read these words [i.e., your brain!] bears little resemblance to a PC. Perhaps our view of computation is too limited. *What if computers were ubiquitous and could be found in many forms? Could a liquid computer exist in which interacting molecules perform computations?* The answer is yes. This is the story of the DNA computer. (Adleman 1998, p. 54; my italics and bracketed interpolation)

Of course, Adleman is not making the Searlean claim that everything is a computer and that, therefore, the interacting molecules of (any) liquid perform computations. Nor is he making the Ballardian claim that DNA computes in the way that a table computes. (Others have, however, made such a claim, on the grounds that strands of DNA are similar to TM tapes with a four symbols instead of two and with

the processes of DNA transcription and recombination as being computable processes (Shapiro and Benenson 2006).) Rather, Adleman's claim is that one can use DNA "to solve mathematical problems". However, contrary to what the editors of *Scientific American* wrote in their subtitle to Adleman's article, it is unlikely that "is redefining what is meant by 'computation' ". After all, the advent of transistors did not change Turing's mathematical characterization of computing any more than the use of vacuum tubes did. At most, DNA computers might change what the lay public means by 'computer'. But that has already happened, with the meaning changing from "humans" to "computing machines"—recall the first *OED* definition from Sect. 2.1 (and see the discussion in Rapaport 2018a, Ch. 6).

Let's take stock of where we are. Presumably, computers are things that compute. Computing is the process that TMs give a precise description for. I.e., computing is what TMs do. And, what TMs do is to move around in a discrete fashion and print discrete marks on discrete sections of the space in which they move around. So, a computer is a *device*—presumably, a *physical* device—that does that. Searle agrees that computing is what TMs do, and he seems to agree that computers are devices that compute. He also believes that everything is a computer; more precisely, he believes that everything can be *described as* a computer (because that's what it means to *be* a computer). And we've also seen reason to think that he might be wrong about that last point.

In the next two sections, we look at two other views about what a computer is.

4.2 Hayes: Computers as Magic Paper

Let's keep straight about three intertwined issues that we have been looking at:

1. What is a computer?
2. Is the brain a computer?
3. Is everything a computer?

Our principal concern is with the first question. Once we have an answer to that, we can try to answer the others. As we've just seen, Searle thinks that a computer is anything that is (or can be described as) a TM, that everything is (or can be described as) a computer, and, therefore, that the brain is a computer, but only trivially so, and not in any interesting sense.

Patrick J. Hayes gives a different definition, in fact, two of them (Hayes 1997). Here's the first:

Definition H1 By "computer" I mean *a machine which performs computations, or which computes*. (Hayes 1997, p. 390; my italics)

A full understanding of this requires a definition of 'computation'; this will be clarified in his second definition. But there are two points to note about this first one.

He prefaces it by saying:

> First, I take it as simply obvious both that computers exist and that not everything is a computer, so that, contra Searle, the concept of "computer" is not vacuous. (Hayes 1997, p. 390)

So, there are (1) things that *are* machines-that-compute, and there are (2) things that are *not* machines-that-compute. Note that (2) can be true in two ways: There might be (2a) *machines* that *don't* compute, or there might be (2b) *things* that *do* compute but that *aren't machines*. Searle disputes the first possibility, because he thinks that everything (including, therefore, any machine) computes. But contrary to what Hayes says, Searle would probably *agree* with the second possibility, because, after all, he thinks that everything (including, therefore, anything that is not a machine) computes! Searle's example of the wall that implements (or that can be interpreted as implementing) Wordstar would be such a non-machine that computes. So, for Hayes's notion to contradict Searle, it must be that Hayes believes that there are machines that do not compute. Perhaps that wall is one of them, or perhaps a dishwasher is a machine that doesn't compute anything.[26]

Are Hayes's two "obvious" points to be understood as criteria of adequacy for any definition—criteria that Hayes thinks need no argument (i.e., as something like "axioms")? Or are they intended to be more like "theorems" that follow from his first definition? If it's the former, then there is no interesting debate between Searle and Hayes; one simply denies what the other argues for. If it's the latter, then Hayes needs to provide arguments or examples to support his position.

A second thing to note about Hayes's definition is that he says that a computer "*performs* computations", not "*can* perform computations". Strictly speaking, your laptop when it is turned off is not a computer by this definition, because it is not performing any computation. And, as Hayes observes,

> On this understanding, a Turing machine is not a computer, but a mathematical abstraction of a certain kind of computer. (Hayes 1997, p. 390)

What about Searle's wall that implements Wordstar? There are two ways to think about how the wall might implement Wordstar. First, it might do so *statically*, simply in virtue of there being a way to map every part of the Wordstar program to some aspect of the molecular or subatomic structure of the wall. In that case, Hayes could well argue that the wall is *not* a Wordstar computer, because it is not computing (even if it *might be able to*). But the wall might implement Wordstar *dynamically*; in fact, that is why Searle thinks that the wall implements Wordstar …

> … because there is some pattern of molecule *movements* which is isomorphic with the formal structure of Wordstar. (Searle 1990, p. 27; my italics)

But a pattern of *movements* suggests that Searle thinks that the wall *is computing*, so it *is* a computer!

Hayes's second definition is a bit more precise, and it is, presumably, his "official" one:

[26] A dishwasher might, however, be described by a (non-computable?) function that takes dirty dishes as input and that returns clean ones as output. The best and most detailed study of what it means for a *machine* to compute is Piccinini (2015). See also Bacon (2010).

Definition H2 [F]ocus on the memory. A computer's memory contains patterns … which are stable but labile [i.e., changeable], and it has the rather special property that changes to the patterns are under the control of other patterns: that is, some of them describe changes to be made to others; and when they do, the memory changes those patterns in the way described by the first ones. … A computer is *a machine which is so constructed that patterns can be put in it, and when they are, the changes they describe will in fact occur to them.* If it were paper, it would be "magic paper" on which writing might spontaneously change, or new writing appear. (Hayes 1997, p. 393; my italics and bracketed interpolation)

There is a subtle difference between Hayes's two definitions, which highlights an ambiguity in Searle's presentation. Recall the distinction between a TM and a UTM: Both TMs and UTMs are hardwired and compute only a single function. The TM computes whichever function is encoded in its machine table; it cannot compute anything else. But the one function, hardwired into its machine table, that a UTM computes is the fetch-execute function that takes as input a program and its data, and that outputs the result of executing that program on that data. In that way, a UTM (besides computing the fetch-execute cycle) can (in a different way) compute any computable function as long as a TM program for that function is encoded and stored on the UTM's tape. The UTM is programmable in the sense that the input program can be varied, not that its hardwired program can be.

Definition H1 seems to include physical TMs (but, as Hayes noted, not abstract ones), because, after all, they compute (at least, when they are turned on and running). Definition H2 seems to *exclude* them, because the second definition requires patterns that describe changes to other patterns. That first kind of pattern is a stored program; the second kind is the data that the program operates on. So, *Definition H2 is for a UTM.*

Here is the ambiguity in Searle's presentation: Is Searle's wall a TM or a UTM? On Searle's view, Wordstar is a TM, so the wall must be a TM, too. So, the wall is not a computer on Definition H2. Could a wall (or a rock, or some other suitably large or complex physical object other than something like a PC or a Mac) be a *universal* TM? My guess is that Searle would say "yes", but it is hard to see how one would actually go about programming it.

The "magic paper" aspect of Definition H2 focuses, as Hayes notes, on the memory, i.e., on the tape. It is as if you were looking at a UTM, but all you saw was the tape, not the read-write head or its states (*m*-configurations; Turing 1936, p. 251)[27] or its mechanism. If you watch the UTM compute, looking only at the tape, you would see the patterns (the '0's and '1's on the tape) "magically" change. A slightly different version of the "magic paper" idea is due to Alan Kay:

> Matter can hold and interpret and act on descriptions that describe anything that matter can do. (Guzdial and Kay 2010)

[27] See Rapaport (2018a, §8.8.2.8.1) for discussion.

The idea of a computer as magic paper or magic matter may seem a bit fantastic. But there are more down-to-earth ways of thinking about this. Richmond Thomason has said that

> … all that a program can do between receiving an input and producing an output is to change variable assignments … (Thomason 2003, p. 328)

A similar point is made by Leslie Lamport:

> [A]n execution of an algorithm is a sequence of states, where a state is an assignment of values to variables. (Lamport 2011, p. 6)

If programs tell a computer how to change the assignments of values to variables, then a computer is a (physical) device that changes the contents of register cells (the register cells that are the physical implementations of the variables in the program). This is really just another version of Turing's machines, if you consider the tape squares to be the register cells.

Similarly, Stuart C. Shapiro points out that

> a computer is a device consisting of a vast number of connected switches. … [T]he switch settings both determine the operation of the device and can be changed by the operation of the device. (Shapiro 2001, p. 3)[28]

So, a switch is a physical implementation of a TM's tape cell, which can also be "in two states" (i.e., have one of two symbols printed on it) and also has a "memory" (i.e., once a symbol is printed on a cell, it remains there until it is changed). Hayes's magic-paper patterns are just Shapiro's switch-settings or Thomason's and Lamport's variable assignments.

Does this definition satisfy Hayes's two criteria? Surely, such machines exist. I wrote this essay on one of them. And surely not everything is such a machine: At least on the face of it, the stapler on my desk is not such "magic paper". Searle, I would imagine, would say that we might see it as such magic paper if we looked at it closely enough and in just the right way. And so the difference between Searle and Hayes seems to be in how one is supposed to look at candidates for being a computer: Do we look at them as we normally do? In that case, not everything is a computer. Or do we squint our eyes and look at them closely in a certain way? In that case, perhaps we could see that everything could be considered to be a computer. Isn't that a rather odd way of thinking about things?

What about the brain? Is it a computer in the sense of "magic paper" (or magic matter)? If Hayes's "patterns" are understood as patterns of neuron firings, then, because surely some patterns of neuron firings cause changes in other such patterns, I think Hayes would consider the brain to be a computer.

[28] For a nice description of what a switch is in this context, see Samuel (1953, p. 1225). For more on computers as switch-setting devices, see the discussions in Stewart (1994) and Brian Hayes (2007) of how train switches can implement computations. Both of these are also examples of TMs implemented in very different media than silicon (namely, trains)!

4.3 Piccinini: Computers as Digital String Manipulators

In a series of three papers, Gualtiero Piccinini has offered an analysis of what a computer is that is more precise than Hayes's and less universal than Searle's (Piccinini 2007b, c, 2008) (see also Piccinini 2015). It is more precise than Hayes's, because it talks about *how* the magic paper performs its tricks. And it is less universal than Searle's, because Piccinini doesn't think that everything is a computer.

As was the case with Hayes, there are two slightly different definitions to be found in Piccinini's papers[29]:

Definition P1 The mathematical theory of how to generate output strings from input strings in accordance with general rules that apply to all input strings and depend on the inputs (and sometimes internal states) for their application is called computability theory. Within computability theory, the activity of manipulating strings of digits in this way is called computation. *Any system that performs this kind of activity is a computing system properly so called.* (Piccinini 2007b, p. 108; my italics)

Definition P2 *[A]ny system whose correct mechanistic explanation ascribes to it the function of generating output strings from input strings (and possibly internal states), in accordance with a general rule that applies to all strings and depends on the input strings (and possibly internal states) for its application, is a computing mechanism.* The mechanism's ability to perform computations is explained mechanistically in terms of its components, their functions, and their organization. (Piccinini 2007c, p. 516; my italics)

These are almost the same, but there is a subtle difference between them.

4.3.1 Definition P1

Let's begin with Definition P1. It implies that a computer is any "system" (presumably, a physical device, because only something physical can actively "perform" an action) that manipulates strings of digits, i.e., that "generate[s] output strings from input strings in accordance with general rules that apply to all input strings and [that] depend on the inputs (and sometimes internal states) for their application". What kind of "general rule"? Piccinini (2008, p. 37) uses the term 'algorithm' instead of 'general rule'. This is consistent with the view that a computer is a TM, and explicates Hayes's "magic trick" as being an algorithm.

The crucial point, according to Piccinini, is that the inputs and outputs must be strings of digits. This is the significant difference between (digital) computers and "analog" computers: The former manipulate strings of digits; the latter manipulate "real variables". Piccinini explicates the difference between digits and real variables as follows:

[29] A third version will be discussed in Sect. 6, below.

> A *digit* is a particular or a discrete state of a particular, discrete in the sense that it belongs to one (and only one) of a finite number of types. … A *string* of digits is a concatenation of digits, namely, a structure that is individuated by the types of digits that compose it, their number, and their ordering (i.e., which digit token is first, which is its successor, and so on). (Piccinini 2007b, p. 107)[30]

Piccinini (2007c, p. 510) observes that a digit is analogous to a letter of an alphabet, so they are like Turing's symbols that can be printed on a TM's tape. On the other hand,

> real variables are physical magnitudes that (i) *vary* over time, (ii) (are assumed to) take a *continuous range of values* within certain bounds, and (iii) (are assumed to) vary *continuously* over time. Examples of real variables include the rate of rotation of a mechanical shaft and the voltage level in an electrical wire. (Piccinini 2008, p. 48)

So far, so good. Neither Searle nor Hayes should be upset with this characterization.

4.3.2 Definition P2

But Piccinini's second definition adds a curious phrase. This definition implies that a computer is any system "whose correct mechanistic explanation ascribes to it the function of" manipulating digit strings according to algorithms. What is the import of that extra phrase?

It certainly sounds as if this is a weaker definition. In fact, it sounds a bit Searlean, because it sounds as if it is not the case that a computer *is* an algorithmic, digit-string manipulator, but rather that it is anything that can be so *described* by some kind of "mechanistic explanation". And that sounds as if being a computer is something "external" and not "intrinsic".

So let's consider what Piccinini has in mind here. He says:

> Roughly, a mechanistic explanation involves a partition of a mechanism into parts, an assignment of functions and organization to those parts, and a statement that a mechanism's capacities are due to the way the parts and their functions are organized. (Piccinini 2007c, p. 502)

Syntax in its most general sense is the study of the properties of a collection of objects and the relations among them (Rapaport 2017b). If a "mechanism" is considered as a collection of its parts, then Piccinini's notion of a mechanistic explanation sounds a lot like a description of the mechanism's syntax. But syntax, you will recall, is what Searle says is not intrinsic to a system (or a mechanism).

So how is Piccinini going to avoid a Searlean "slippery slope" and deny that everything is a computer? One way he tries to do this is by suggesting that even if a system can be analyzed syntactically in different ways, only one of those ways will help us understand the system's behavior:

[30] In the other two papers in his trilogy, Piccinini gives slightly different characterizations of what a digit is, but these need not concern us here; see Piccinini (2007c, p. 510, 2008 p. 34).

> Mechanistic descriptions are sometimes said to be perspectival, in the sense that the same component or activity may be seen as part of different mechanisms depending on which phenomenon is being explained …. For instance, the heart may be said to be for pumping blood as part of an explanation of blood circulation, or it may be said to be for generating rhythmic noises as part of an explanation of physicians who diagnose patients by listening to their hearts. This kind of perspectivalism does not trivialize mechanistic descriptions. Once we fix the phenomenon to be explained, the question of what explains the phenomenon has an objective answer. This applies to computations as well as other capacities of mechanisms. A heart makes the same noises regardless of whether a physician is interested in hearing it or anyone is interested in explaining medical diagnosis. (Piccinini 2007c, p. 516)

Let's try to apply this to Searle's "Wordstar wall": From one perspective, the wall is just a wall; from another, according to Searle, it can be taken as an implementation of Wordstar. Compare this to Piccinini's claim that, from one perspective, a heart is a pump, and, from another, it is a noisemaker. If you're a doctor interested in hearing the heart's noises, you'll consider the heart as a noisemaker. If you're a doctor interested in making a medical diagnosis, you'll consider it as a pump. Similarly, if you're a house painter, say, you'll consider the wall as a flat surface to be colored, but if you're Searle, you'll try to consider it as a computer program. (Although I don't think you'll be very successful in using it to write a philosophy essay!)

5 What Else Might be a Computer?

So, what is a computer? It would seem that almost all proposed definitions agree on at least the following:

- Computers are physical devices.
- They interact with other physical devices in the world.
- They algorithmically manipulate (physical) symbols (strings of digits), converting some into others.
- They are physical implementations of (U)TMs the sense that their I/O behavior is logically equivalent to that of a (U)TM (even though the details of their processing might not be).[31]

Does such a definition include too much? Let's assume for a moment that something like Piccinini's reply to Searle carries the day, so that it makes sense to say that not everything is a computer. Still, might there be some things that intuitively aren't computers but that turn out to be computers on even our narrow characterization?

[31] A slight modification of this might be necessary to avoid the possibility that a physical device might be considered to be a computer even if it doesn't compute: We probably want to rule out "real magic", for instance Rapaport (2017c, §13.6).

This is always a possibility. Any time that you try to make an informal concept precise, you run the risk of *including* some things under the precise concept that didn't (seem to) fall under the informal concept. You also run the risk of *excluding* some things that did. One way to react to this situation is to reject the formalization, or else to refine it so as to minimize or eliminate the "erroneous" inclusions and exclusions. But another reaction is to bite the bullet and agree to the new inclusions and exclusions (Rapaport 2018a, §3.3.3.1). For instance, you might even come to see that something that you didn't think was a computer really was one.

In this section, we'll consider two things that may—or may not!—turn out to be computers: the brain, and the universe.

5.1 Is a Brain a Computer?

> Many real-world computational systems compute more than just a single function—the world has moved to interactive computing (Goldin et al. 2006). The term reactive system is used to describe a system that maintains an ongoing interaction with its environment. …
> A distributed system is one that consists of autonomous computing systems that communicate with one another through some kind of network using message passing. …
> Perhaps the most intriguing examples of reactive distributed computing systems are biological systems such as cells and organisms. *We could even consider the human brain to be a biological computing system.* Formulation of appropriate models of computation for understanding biological processes is a formidable scientific challenge in the intersection of biology and computer science. (Aho 2011, p. 6; my italics)

Many people claim that the (human) brain is a computer. Searle thinks it is, but only because he thinks that everything is a computer. But perhaps there is a more interesting way in which the brain is a computer. Certainly, contemporary computational cognitive science uses computers as at least a metaphor for the brain.[32]

In fact, "computationalism" is sometimes taken to be the view that the brain (or the mind) *is* a computer, or that the brain (or the mind) computes, or that brain (or mental) states and processes *are* computational states and processes[33]:

> The basic idea of the computer model of the mind is that the mind is the program and the brain the hardware of a computational system. (Searle 1990, p. 21)

[32] Before computers came along, there were many other physical metaphors for the brain: The brain was considered to be like a telephone system or like a plumbing system. See, e.g., Lewis (1953), Squires (1970), Sternberg (1990), Gigerenzer and Goldstein (1996), Angier (2010), Guernsey (2009), Pasanek (2015), and US National Library of Medicine (2015).

[33] For more such sentiments, see Rapaport (2012, §2). On computationalism more generally, see Rescorla (2015).

> The core idea of cognitive science is that our brains are a kind of computer …. Psychologists try to find out exactly what kinds of programs our brains use, and how our brains implement those programs. (Gopnik 2009, p. 43)

> Computationalism … is the view that the functional organization of the brain (or any other functionally equivalent system) is computational, or that neural states are computational states. (Piccinini 2010, p. 271; cf. pp. 277–278)

But if one of the essential features of a computer is that it carries out computable processes by *computing* rather than (say) by some biological but non-computational technique, then it's at least logically possible that the brain is not a computer even if brain processes are computable. Indeed, I prefer to define 'computationalism' as the view that cognition is *computable*, not necessarily that it is *computed* (by a brain) (Rapaport 1998, 2012, 2018b).[34]

How can this be? A process is computable if and only if there is an algorithm (or a system of algorithms) that specifies how that process can be carried out. But it is logically possible for a process to be computable in this sense without actually being computed.

Here are some examples:

1. Someone might come up with a computational theory of the behavior of the stock market, yet the actual stock market's behavior is determined by the individual decisions made by individual investors and not by anyone or anything executing an algorithm. I.e., the behavior might be comput*able* even if it is not comput*ational*.
2. Calculations done by slide rules are done by analog means, yet the calculations themselves are clearly computable. Analog computations are not normally considered to be TM computations.
3. Hayes's magic paper is a logically, if not physically, possible example.
4. Another example might be the brain itself. Piccinini has argued that neuron firings (more specifically, "spike trains"—i.e., sequences of "action potential"—in groups of neurons) are not representable as digit strings (Piccinini 2005, 2007a). But, because Piccinini believes that a device is not a computer unless it manipulates digit strings, and because it is generally considered that human cognition is implemented by neuron firings, it follows that the brain's cognitive functioning—even if comput*able*—is not accomplished by comput*ation*. Yet, if cognitive functions are computable (as contemporary cognitive science suggests—see Edelman 2008a), then there would still be algorithms that compute cognition, even if the brain doesn't do it that way.

David Chalmers puts the point this way:

> Is the brain a [programmable] computer … ? Arguably. For a start, the brain can be "programmed" to implement various computations by the laborious means of conscious serial rule-following; but this is a fairly incidental ability.

[34] And I do not assume that all cognition is computable; instead, one should ask, "How much of cognition is computable?" (Rapaport 1998, p. 405).

> On a different level, it might be argued that learning provides a certain kind of programmability and parameter-setting, but this is a sufficiently indirect kind of parameter-setting that it might be argued that it does not qualify. In any case, the question is quite unimportant for our purposes. *What counts is that the brain implements various complex computations, not that it is a computer.* (Chalmers 2011, §2.2, esp. p. 336, my bracketed interpolation and italics)

There are two interesting points made here. The first is that the brain can simulate a TM "by … conscious serial rule-following": Recall our discussion of Samuel in Sect. 2.3. The second is the last sentence: What really matters is that the brain can have I/O behavior that is computable, not that it "is" a computer. To say that it is a computer raises the question of what kind of computer it is: A Turing machine? A register machine? Something *sui generis*? And these questions seem to be of less interest than the fact that its behavior is computable.

As Ballard (1997, pp. 1–2) puts it, "The key question … is, Is computation sufficient to model the brain?". One reason this is an interesting question is that researchers in vision have wondered "how … an incomplete description, encoded within neural states, [could] be sufficient to direct the survival and successful adaptive behavior of a living system" (Richards 1988, as cited in Ballard 1997, p. 2). If a computational model of this ability is sufficient, then it might also be sufficient to model the brain. And this might be the case even if, as, e.g., Piccinini and Bahar (2013) argue, the brain itself is *not* a computer, i.e., does not behave in a computational fashion.[35] A model of a phenomenon does not need to be identical in all respects to the phenomenon that it models, as long as it serves the purposes of the modeling. But Ballard also makes the stronger claim when he says, a few pages later, "*If the brain is performing computation*, it should obey the laws of computational theory" (Ballard 1997, p. 6, my italics). But whether the brain performs computations is a different question from whether its performance can be modeled or described in computational terms. So, the brain doesn't have to *be* a computer in order for its behavior to be describable computationally. As Churchland and Sejnowski (1992) note, whether the brain *is* a computer—whether, that is, the brain's functioning satisfies one of the (logically equivalent) characterizations of computing—is an empirical issue.

Still, if the brain computes in some way (or "implements computations"), and if a computer is, by definition, something that computes, then we might still wonder if the brain is some kind of computer. As I once read somewhere, "The best current explanation of how a brain could instantiate this kind of system of rules and representations is that it is a kind of computer." Thus, we have here the makings of an abductive argument that the brain is a computer. Note that this is a much more reasonable argument than Searle's or than trying to model the brain as, say, a TM.[36] And, as Marcus (2015) observes, "For most neuroscientists, this is just a bad

[35] For some other arguments that the brain is *not* a computer, see Naur (2007, p. 85), Schulman (2009), Linker (2015).

[36] It is one thing to argue that brains are (or are not) computers of some kind. It is quite another to argue that they are TMs in particular. The earliest suggestion to that effect is McCulloch and Pitts (1943). For a critical and historical review of that classic paper, see Piccinini (2004). More recently, the cognitive

metaphor. But it's still the most useful analogy that we have. … The sooner we can figure out what kind of computer the brain is, the better."

If the brain *is* a computer, then it is a fine example of a *biological* computer. We have already looked at the possibility of biological computers implemented in DNA (Sect. 4.1.7, above). Whether there might be other kinds of biological computers is in part an empirical question (and in part a function of how we define 'computer').[37] But there seems to be no reason why the implementing mechanism for a UTM couldn't be biological. On the other hand, to the extent that, as one referee observed, biological evolution has produced any "natural" computers (including brains), there is no reason to think that they must be equivalent to TMs (https://www.cs.york.ac.uk/nature/gc7/summary.pdf).

5.2 Is the Universe a Computer?

> A computation is a process that establishes a mapping among some symbolic domains. … Because it involves symbols, this definition is very broad: a system instantiates a computation if its dynamics can be interpreted (by another process) as establishing the right kind of mapping.
> Under this definition, a stone rolling down a hillside computes its position and velocity in exactly the same sense that my notebook computes the position and the velocity of the mouse cursor on the screen (they just happen to be instantiating different symbolic mappings). Indeed, the universe in its entirety also instantiates a computation, albeit one that goes to waste for the lack of any process external to it that would make sense of what it is up to. (Edelman 2008b, pp. 182–183)

Might the universe itself be a computer?[38] Consider Kepler's laws of planetary motion. Are they just a computable theory that describes the behavior of the solar system? If so, then a computer that calculates with them might be said to *simulate* the solar system in the same way that any kind of program might be said to simulate a physical (or biological, or economic) process, or in the same way that an AI program might be said to simulate a cognitive process.

Or does *the solar system itself* compute Kepler's laws? If so, then the solar system would seem to be a (special purpose) computer (i.e., a kind of TM). After all, if "*biological computation* is a process that occurs in nature, not merely in computer simulations of nature" (Mitchell 2011, p. 2), then it is at least not unreasonable that the solar system computes Kepler's Laws:

> Going further along the path of nature, suppose that we have a detailed mathematical model of some physical process such as—say—a chemical

Footnote 36 continued
neuroscientist Stanislas Dehaene and his colleagues have made similar arguments; see Sackur and Dehaene (2009), Zylberberg et al. (2011).

[37] For a survey, see https://en.wikipedia.org/wiki/Biological_computing.

[38] For humorous illustrations of this, see the fake Google search page at http://abstrusegoose.com/115 and the cartoon at http://abstrusegoose.com/219.

> reaction; clearly we can either organise the reaction in the laboratory and observe the outcome, or we can set up the mathematical model of the reaction on a computer either as the numerical solution of a system of equations, or as a Montecarlo simulation, and we can then observe the outcome. We can all agree that when we "run the reaction" on the computer either as a numerical solution or a Montecarlo simulation, we are dealing with a computation.
> But why then not also consider that the laboratory experiment itself is after all only a "computational analogue" of the numerical computer experiment! In fact, the laboratory experiment will be a mixed analogue and digital phenomenon because of the actual discrete number of molecules involved, even though we may not know their number exactly. In this case, the "hardware" used for the computation are the molecules and the physical environment that they are placed in, while the software is also inscribed in the different molecules species that are involved in the reaction, via their propensities to react with each other …. (Gelenbe 2011, pp. 3–4)

As we just saw in the case of the brain, there might be a computational *theory* of some phenomenon (e.g., Kepler's laws)—i.e., the phenomenon might be *computable*—but the phenomenon *itself* need not be produced computationally.

> Indeed, *computational algorithms are so powerful that they can simulate virtually any phenomena, without proving anything about the computational nature of the actual mechanisms underlying these phenomena.* Computational algorithms generate a perfect description of the rotation of the planets around the sun, although the solar system does not compute in any way. In order to be considered as providing a model of the mechanisms actually involved, and not only a simulation of the end-product of mechanisms acting at a different level, computational models have to perform better than alternative, noncomputational explanations. (Perruchet and Vinter 2002, §1.3.4, p. 300; my italics)

Nevertheless, could it be the case that our solar system *is* computing Kepler's laws? Arguments along these lines have been put forth by Stephen Wolfram and by Seth Lloyd.

5.2.1 Wolfram's Argument

Stephan Wolfram (2002b) argues as follows:

1. Nature is discrete.
2. Therefore, possibly it is a cellular automaton.
3. There are cellular automata that are equivalent to a TM.
4. Therefore, possibly the universe is a computer.

There are a number of problems with this argument. First, why should we believe that nature (i.e., the universe) is discrete? Presumably, because quantum mechanics says that it is. But some distinguished physicists deny this (Weinberg 2002). So, at best, for those of us who are not physicists able to take a stand on this issue,

Wolfram's conclusion has to be conditional: If the universe is discrete, then possibly it is a computer.

But let's suppose (for the sake of the argument) that nature *is* discrete. Might it be a "cellular automaton"?[39] But, of course, even if a discrete universe *might* be a cellular automaton, it *need not* be. If it isn't, the argument stops here. But, if it is, then—because the third premise is mathematically true[40]—the conclusion follows validly from the premises. Premise 2 is the one most in need of justification. But even if all of the premises and (hence) the conclusion are true, it is not clear what philosophical consequences we are supposed to draw from this.[41]

5.2.2 Lloyd's Argument

Seth Lloyd also argues that the universe is a computer because nature is discrete, but Lloyd's intermediate premises differ from Wolfram's. Lloyd argues as follows (Lloyd and Ng 2004):

1. Nature is discrete. (This is "the central maxim of quantum mechanics" (p. 54).)
2. In particular, elementary particles have a "spin axis" that can be in one of two directions.
3. ∴ They encode a bit.
4. ∴ Elementary particles store bits of information.
5. Interactions between particles can flip the spin axis; this transforms the stored data—i.e., these interactions are operations on the data.
6. ∴ (Because any physical system stores and processes information,) all physical systems are computers.
7. In particular, a rock is a computer.
8. Also, the entire universe is a computer.

Premise 1 matches Wolfram's fundamental premise and would seem to be a necessity for anything to be considered a digital computer. The next four premises also underlie quantum computing.

But the most serious problem with Lloyd's argument as presented here is premise 6. Is the processing sufficient to be considered to be TM-equivalent computation? Perhaps; after all, it seems that all that is happening is that cells change from 0s to 1s and vice versa. But that's not all that's involved in computing. (Or is it? Isn't that what Hayes's magic-paper hypothesis says?) What about the control structures—the grammar—of the computation?

[39] The easiest way to think of a cellular automaton is as a two-dimensional TM tape for which the symbol in any cell is a function of the symbols in neighboring cells (https://en.wikipedia.org/wiki/Cellular_automaton). On cellular automata, see Burks (1970).

[40] See, e.g., https://en.wikipedia.org/wiki/Turing_completeness, https://en.wikipedia.org/wiki/Rule_110, and https://en.wikipedia.org/wiki/Conway%27s_Game_of_Life.

[41] For more on Wolfram, see http://www.stephenwolfram.com/ and Wolfram (2002a). For a critical review, see Weinberg (2002). Aaronson (2011) claims that quantum computing has "overthrown" views like those of Wolfram (2002b) that "the universe itself is basically a giant computer … by showing that if [it is, then] it's a vastly more powerful kind of computer than any yet constructed by humankind."

And although Lloyd wants to conclude that everything in the universe (including the universe itself!) is a computer, note that this is not exactly the same as Searle's version of that claim. For Searle, everything can be interpreted as *any* computer program. For Lloyd, anything is a computer, "although they may not accept input or give output in a form that is meaningful to humans" (p. 55). So, for Lloyd, it's not a matter of interpretation. Moreover, "analyzing the universe in terms of bits and bytes does not replace analyzing it in conventional terms such as force and energy" (p. 54). It's not clear what the import of that is: Does he mean that the computer analysis is irrelevant? Probably not: "it does uncover new and surprising facts" (p. 54), though he is vague on what those "facts" are. Does he mean that there are different ways to understand a given object? E.g., an object could be understood as a computer subject to the laws of computability theory or as a physical object subject to the laws of physics. That is true, but unsurprising: Animals, for instance, can be understood as physical objects subject to the laws of physics as well as being understood as biological objects subject to the laws of biology. Does he mean that force and energy can, or should, be understood in terms of the underlying computational nature of physical objects? He doesn't say.

But Lloyd does end with a speculation on what it is that the universe is computing, namely, itself! Or, as he puts it, "computation is existence" (p. 61). As mystical as this sounds, does it mean anything different from the claim that the solar system computes Kepler's Law?[42]

Finally, here's an interesting puzzle for Lloyd's view, relating it to issues concerning whether a computer must halt.

> [A]ssuming the universe is computing its own evolution …, does it have a finite lifetime or not? If it is infinite, then its self-computation won't get done; it never produces an answer …. Hence, it does not qualify as a computation. (Borbely 2005, p. 15)

Turing, however, might not have considered this to be a problem: His original *a*-machines only computed the decimal expansions of real numbers by *not* halting! (They were "circle-free" (Turing 1936, §2); for discussion, see Rapaport 2018a, §8.10.3.1.)

6 Conclusion

So, finally, what is a computer?

At a bare minimum, we should probably say that a (programmable) computer is a physically plausible implementation (including a virtual implementation) of anything logically equivalent to a UTM (our DC4, above). Most of the definitions that we discussed above might best be viewed as focusing on exactly what is meant by 'implementation' or which entities count as such implementations.

[42] For more on Lloyd, see Lloyd (2000, 2002, 2006), Powell (2006), Schmidhuber (2006). Computer pioneer Konrad Zuse also argued that the universe is a computer (Schmidhuber 2002). For related views, see Bostrom (2003), Chaitin (2006), Bacon (2010), Hidalgo (2015), O'Neill (2015).

Two kinds of (alleged) computers are not obviously included in this sort of definition: analog computers and "hypercomputers". Because most discussions focus on "digital" computers as opposed to analog ones, I have not considered analog computers here. By 'hypercomputer', I have in mind any physical implementation (assuming that there are any) of anything capable of "hypercomputation", i.e., anything capable of "going beyond the Turing limit", i.e., anything that "violates" the Church–Turing Computability Thesis.

The topics of hypercomputation and counterexamples to the Computability Thesis are beyond our scope (but see Rapaport 2018a, Ch. 11 for discussion). But one way to incorporate these other models of computation into a unified defintion of 'computer' might be this:

(DC5) A computer is any physically plausible implementation of anything that is *at least* logically equivalent to a UTM.

In other words, if something can compute at least all TM-computable functions, but might also be able to perform analog computations or hypercomputations, then it, too, is a computer. A possible objection to this is that an adding machine, or a calculator, or a machine that is designed to do only *sub*-Turing computation, such as a physical implementation of a finite automaton, has at least some claim to being called a 'computer'.

So another way to incorporate all such models is to go one step beyond our DC5 to:

(DC6) A computer is a physically plausible implementation of some model of computation.

Indeed, Piccinini (2018, p. 2) has more recently offered a definition along these lines. He defines 'computation' as "the processing of medium independent vehicles by a functional mechanism in accordance with a rule." (See Piccinini 2015, Ch. 7 for argumentation and more details.) This, of course, is a definition of 'computation', not 'computer'. But we can turn it inside out to get this:

Definition P3 A computer is a functional mechanism that processes medium-independent vehicles in accordance with a rule.

He explicitly cites as an advantage of this very broad definition its inclusion of "not only digital but also analog and other unconventional types of computation" (p. 3)—including hypercomputation. But Piccinini (2015, Chs. 15 and 16) also distinguishes between the "mathematical" Church–Turing Computability Thesis and a "modest physical" thesis: "Any function that is physically computable is Turing-computable" (Piccinini 2015, p. 264), and he argues that it is an "open empirical question" (p. 273) whether hypercomputers are possible (although he doubts that they are).

My only hesitations with my DC6 and Piccinini's P3 are that they seem to be a bit too vague in their generosity, leaving all the work to the meaning of computation. But maybe that's exactly right. Despite its engineering history and despite its name, perhaps "computer science" is best viewed as the scientific study of computation, not (just) computers. Or, as I have said elsewhere, "computer science is the (scientific …) study of what problems can be solved, what tasks can be accomplished, and what features of the world can be understood computationally … and then to provide algorithms to show how this can be done efficiently, practically, physically, and ethically" (Rapaport 2017c, §15, p. 16). Determining how computation can be done *physically* tells us what a computer is.

References

Aaronson, S. (2011, December 9). Quantum computing promises new insights, not just supermachines. *New York Times*, D5. http://www.nytimes.com/2011/12/06/science/scott-aaronson-quantum-computing-promises-new-insights.html.

Adleman, L. M. (1998). Computing with DNA. *Scientific American*, 54–61. http://www.usc.edu/dept/molecular-science/papers/fp-sciam98.pdf.

Aho, A. V. (2011, January). What is computation? Computation and computational thinking. *Ubiquity*, Article 1. http://ubiquity.acm.org/article.cfm?id=1922682.

Aizawa, K. (2010). Computation in cognitive science: It is not all about Turing-equivalent computation. *Studies in History and Philosophy of Science*, *41*(3), 227–236. Reply to Piccinini (2004).

Anderson, D. L. (2006). The nature of computers. *The Mind Project*. http://www.mind.ilstu.edu/curriculum/modOverview.php?modGUI=196.

Angier, N. (2010, February 2). Abtract thoughts? The body takes them literally. *New York Times/Science Times*, D2. http://www.nytimes.com/2010/02/02/science/02angier.html.

Bacon, D. (2010, December). What is computation? Computation and fundamental physics. *Ubiquity 2010*(December). Article 4, http://ubiquity.acm.org/article.cfm?id=1920826.

Bader, R. M. (2013). Towards a hyperintensional theory of intrinsicality. *Journal of Philosophy*, *110*(10), 525–563.

Ballard, D. H. (1997). *An introduction to natural computation*. Cambridge, MA: MIT Press.

Benacerraf, P. (1965). What numbers could not be. *Philosophical Review*, *74*(1), 47–73.

Biermann, A. (1990). *Great ideas in computer science: A gentle introduction*. Cambridge, MA: MIT Press.

Blum, L., Shub, M., & Smale, S. (1989). On a theory of computation and complexity over the real numbers: NP-completeness, recursive functions, and universal machines. *Bulletin of the American Mathematical Society*, *21*(1), 1–46.

Borbely, R. (2005). Letter to the editor. *Scientific American*, *12*, 15.

Bostrom, N. (2003). Are you living in a computer simulation? *Philosophical Quarterly*, *53*(211), 243–255.

Burks, A. W. (Ed.). (1970). *Essays on cellular automata*. Urbana, IL: University of Illinois Press.

Buzen, J. P. (2011, January). Computation, uncertainty and risk. *Ubiquity*, Article 5. http://ubiquity.acm.org/article.cfm?id=1936886.

Care, C. (2007). Not only digital: A review of ACM's early involvement with analog computing technology. *Communications of the ACM*, *50*(5), 42–45.

Carpenter, B., & Doran, R. (1977). The other Turing machine. *The Computer Journal*, *20*(3), 269–279.

Chaitin, G. (2006). How real are real numbers? *International Journal of Bifurcation and Chaos*. http://www.cs.auckland.ac.nz/~chaitin/olympia.pdf (2006 version); http://www.umcs.maine.edu/~chaitin/wlu.html (2009 version).

Chalmers, D. J. (1996). Does a rock implement every finite-state automaton? *Synthese*, *108*, 309–333. http://consc.net/papers/rock.html; Chalmers corrects "an error in my arguments" in Chalmers (2012, pp. 236–238).

Chalmers, D. J. (2011). A computational foundation for the study of cognition. *Journal of Cognitive Science (South Korea)*, *12*(4), 323–357.
Chalmers, D. J. (2012). The varieties of computation: A reply. *Journal of Cognitive Science (South Korea)*, *13*(3), 211–248.
Chirimuuta, M., Boone, T., & DeMedonsa, M. (2014, September 19). Is your brain a computer? (video). *Instant HPS*. https://www.youtube.com/watch?v=-8q_UXpHsaY.
Chow, S. J. (2015). Many meanings of 'heuristic'. *British Journal for the Philosophy of Science*, *66*, 977–1016.
Churchland, P. S., & Sejnowski, T. J. (1992). *The computational brain*. Cambridge, MA: MIT Press.
Coffa, J. (1991). *The semantic tradition from Kant to Carnap: To the Vienna Station*. Cambridge: Cambridge University Press.
Copeland, B. J. (1996). What is computation? *Synthese*, *108*, 335–359.
Copeland, B. J. (1997). The broad conception of computation. *American Behavioral Scientist*, *40*(6), 690–716.
Copeland, B. J. (2002). Hypercomputation. *Minds and Machines*, *12*(4), 461–502.
Copeland, B. J. (Ed.). (2004). *The essential Turing*. Oxford: Oxford University Press.
Copeland, B. J. (2013, August 12). What Apple and Microsoft owe to Turing. *Huff[ington] Post Tech/The Blog*. http://www.huffingtonpost.com/jack-copeland/what-apple-and-microsoft-_b_3742114.html.
Corry, L. (2017). Turing's pre-war analog computers: The fatherhood of the modern computer revisited. *Communications of the ACM*, *60*(8), 50–58.
Davis, M. D. (2000). Overheard in the park. *American Scientist*, *88*, 366–367.
Davis, M. D. (2004). The myth of hypercomputation. In C. Teuscher (Ed.), *Alan Turing: The life and legacy of a great thinker* (pp. 195–212). Berlin: Springer.
Davis, M. D. (2006a). The Church–Turing thesis: Consensus and opposition. In A. Beckmann, U. Berger, B. Löwe, & J. Tucker (Eds.), *Logical approaches to computational barriers: Second conference on computability in Europe, CiE 2006, Lecture notes in computer science* (Vol. 3988, pp. 125–132), Swansea, UK, June 30–July 5, 2006. Berlin: Springer.
Davis, M. D. (2006b). Why there is no such discipline as hypercomputation. *Applied Mathematics and Computation*, *178*, 4–7.
Daylight, E. G. (2013). *Towards a historical notion of "Turing—The father of computer science"*. http://www.dijkstrascry.com/sites/default/files/papers/Daylightpaper91.pdf.
Dennett, D. C. (2013). *Intuition pumps and other tools for thinking*. New York: W.W. Norton.
Dennett, D. C. (2017). *From bacteria to bach and back: The evolution of mind*. New York: W.W. Norton.
Edelman, S. (2008a). *Computing the mind*. New York: Oxford University Press.
Edelman, S. (2008b). On the nature of minds; or: Truth and consequences. *Journal of Experimental & Theoretical Artificial Intelligence*, *20*(3), 181–196.
Egan, F. (2012). Metaphysics and computational cognitive science: Let's not let the tail wag the dog. *Journal of Cognitive Science (South Korea)*, *13*(1), 39–49.
Fortnow, L. (2010, December). What is computation? *Ubiquity*, Article 5. http://ubiquity.acm.org/article.cfm?id=1921573.
Gelenbe, E. (2011, February). Natural computation. *Ubiquity*, Article 1. http://ubiquity.acm.org/article.cfm?id=1940722.
Gigerenzer, G., & Goldstein, D. G. (1996). Mind as computer: Birth of a metaphor. *Creativity Research Journal*, *9*(2–3), 131–144.
Gödel, K. (perhaps about 1938). Undecidable diophantine propositions. In S. Feferman, et al. (Eds.), *Collected works* (Vol. III, pp. 164–175). Oxford: Oxford University Press.
Goldin, D., Smolka, S. A., & Wegner, P. (Eds.). (2006). *Interactive computation: The new paradigm*. Berlin: Springer.
Goodman, N. D. (1987). Intensions, Church's thesis, and the formalization of mathematics. *Notre Dame Journal of Formal Logic*, *28*(4), 473–489.
Gopnik, A. (2009). *The philosophical baby: What children's minds tell us about truth, love, and the meaning of life*. New York: Farrar, Straus and Giroux.
Guernsey, L. (2009). Computers track the elusive metaphor. *Chronicle of Higher Education*, *55*, A11.
Guzdial, M., & Kay, A. (2010, May 24). The core of computer science: Alan Kay's 'triple whammy'. *Computing Education Blog*. http://computinged.wordpress.com/2010/05/24/the-core-of-computer-science-alan-kays-triple-whammy/.

Haigh, T. (2013). 'Stored program concept' considered harmful: History and historiography. In P. Bonizzoni, V. Brattka, & B. Löwe (Eds.), *CiE 2013, Lecture notes in computer science* (Vol. 7921, pp. 241–251). Berlin: Springer.

Harnish, R. M. (2002). Coda: Computation for cognitive science, or what IS a computer, anyway? In *Minds, brains, computers: An historical introduction to the foundations of cognitive science* (pp. 394–412). Malden, MA: Blackwell. https://www.amazon.com/Minds-Brains-Computers-Introduction-Foundations/dp/0631212604/.

Hartree, D. (1949). *Calculating instruments and machines*. Urbana, IL: University of Illinois Press.

Hayes, B. (2007). Trains of thought. *American Scientist*, *95*(2), 108–113.

Hayes, P. J. (1997). What is a computer? *Monist*, *80*(3), 389–404.

Hedger, L. (1998). *Analog computation: Everything old is new again* (Vol. 21). Indiana University Research & Creative Activity. http://www.indiana.edu/~rcapub/v21n2/p24.html.

Hidalgo, C. (2015). Planet hard drive. *Scientific American*, *313*(2), 72–75.

Hintikka, J., & Mutanen, A. (1997). An alternative concept of computability. In J. Hintikka (Ed.), *Language, truth, and logic in mathematics* (pp. 174–188). Dordrecht: Springer.

Hogarth, M. (1992). Does general relativity allow an observer to view an eternity in a finite time? *Foundations of Physics Letters*, *5*(2), 173–181.

Holst, P. A. (2000). Analog computer. In A. Ralston, E. D. Reilly, & D. Hemmendinger (Eds.), *Encyclopedia of computer science* (4th ed., pp. 53–59). New York: Grove's Dictionaries.

Jackson, A. S. (1960). *Analog computation*. New York: McGraw-Hill.

Kanat-Alexander, M. (2008, October 10). What is a computer? *Code Simplicity*. http://www.codesimplicity.com/post/what-is-a-computer/.

Kleene, S. C. (1995). Turing's analysis of computability, and major applications of it. In R. Herken (Ed.), *The universal turing machine: A half-century survey* (2nd ed., pp. 15–49). Vienna: Springer.

Kugel, P. (2002). Computing machines can't be intelligent (… and Turing said so). *Minds and Machines*, *12*(4), 563–579.

Lamport, L. (2011). Euclid writes an algorithm: A fairytale. International Journal of Software and Informatics 5(1–2, Part 1), 7–20.

Langton, R., & D. Lewis (1998). Defining 'intrinsic'. *Philosophy and Phenomenological Research*, *58*(2), 333–345.

Lewis, D. (1983). Extrinsic properties. *Philosophical Studies*, *44*(2), 197–200.

Lewis, W. (1953). Electronic computers and telephone switching. *Proceedings of the Institute of Radio Engineers*, *41*(10), 1242–1244.

Linker, D. (2015, July 1). No, your brain isn't a computer. *The Week*. http://theweek.com/articles/563975/no-brain-isnt-computer.

Lloyd, S. (2000). Ultimate physical limits to computation. *Nature*, *406*, 1047–1054.

Lloyd, S. (2002). Computational capacity of the universe. *Physical Review Letters*, *88*(23), 237901-1–237901-4.

Lloyd, S. (2006). *Programming the universe: A quantum computer scientist takes on the cosmos*. New York: Alfred A. Knopf.

Lloyd, S., & Ng, Y. J. (2004). Black hole computers. *Scientific American*, *291*(5), 52–61.

Marcus, G. (2015, June 28). Face it, your brain is a computer. *New York Times Sunday Review*, SR12. http://www.nytimes.com/2015/06/28/opinion/sunday/face-it-your-brain-is-a-computer.html.

Marshall, D. (2016). The varieties of intrinsicality. *Philosophy and Phenomenological Research*, *92*(2), 237–263.

McCulloch, W. S., & Pitts, W. H. (1943). A logical calculus of the ideas immanent in nervous activity. *Bulletin of Mathematical Biophysics*, *7*, 114–133. Reprinted in Warren S. McCulloch, *Embodiments of mind* (Cambridge, MA: MIT Press, 1965): 19–39.

McMillan, R. (2013, July 7). The end of digital tyranny: Why the future of computing is analog. *Wired*. http://www.wired.com/wiredenterprise/2013/07/analogfuture/.

Mitchell, M. (2011, February). What is computation? Biological computation. *Ubiquity*, Article 3. http://ubiquity.acm.org/article.cfm?id=1944826.

Moor, J. H. (1978). Three myths of computer science. *British Journal for the Philosophy of Science*, *29*(3), 213–222.

Morris, G. J., & Reilly, E. D. (2000). Digital computer. In A. Ralston, E. D. Reilly, & D. Hemmendinger (Eds.), *Encyclopedia of computer science* (4th ed., pp. 359–545). New York: Grove's Dictionaries.

Naur, P. (2007). Computing versus human thinking. *Communications of the ACM*, *50*(1), 85–94.

O'Neill, S. (2015). The human race is a computer. *New Scientist*, *227*(3029), 26–27. Interview with César Hidalgo.

Oommen, B. J., & Rueda, L. G. (2005). A formal analysis of why heuristic functions work. *Artificial Intelligence*, *164*(1–2), 1–22.

Pasanek, B. (2015). The mind is a metaphor. http://metaphors.iath.virginia.edu/.

Perruchet, P., & Vinter, A. (2002). The self-organizing consciousness. *Behavioral and Brain Sciences*, *25*(3), 297–388.

Piccinini, G. (2004). The first computational theory of mind and brain: A close look at McCulloch and Pitts's 'Logical calculus of ideas immanent in nervous activity'. *Synthese*, *141*, 175–215. For a reply, see Aizawa (2010).

Piccinini, G. (2005). *Symbols, strings, and spikes: The empirical refutation of computationalism*. Abstract at https://philpapers.org/rec/PICSSA; unpublished paper at https://web.archive.org/web/20080216023546/http://www.umsl.edu/~piccininig/Symbols%20Strings%20and%20Spikes%2019.htm. superseded by Piccinini and Bahar (2013).

Piccinini, G. (2006). Computational explanation in neuroscience. *Synthese*, *153*(3), 343–353.

Piccinini, G. (2007a). Computational explanation and mechanistic explanation of mind. In M. Marraffa, M. D. Caro, & F. Ferretti (Eds.), *Cartographies of the mind: Philosophy and psychology in intersection* (pp. 23–36). Dordrecht: Springer.

Piccinini, G. (2007b). Computational modelling vs. computational explanation: Is everything a Turing machine, and does it matter to the philosophy of mind? *Australasian Journal of Philosophy*, *85*(1), 93–115.

Piccinini, G. (2007c). Computing mechanisms. *Philosophy of Science*, *74*(4), 501–526.

Piccinini, G. (2008). Computers. *Pacific Philosophical Quarterly*, *89*, 32–73.

Piccinini, G. (2010). The mind as neural software? Understanding functionalism, computationalism, and computational functionalism. *Philosophy and Phenomenological Research*, *81*(2), 269–311.

Piccinini, G. (2011). The physical Church–Turing thesis: Modest or bold? *British Journal for the Philosophy of Science*, *62*, 733–769.

Piccinini, G. (2015). *Physical computation: A mechanistic account*. Oxford: Oxford University Press.

Piccinini, G. (2018). Computation and representation in cognitive neuroscience. *Minds and Machines*, *28*(1), 1–6.

Piccinini, G., & Bahar, S. (2013). Neural computation and the computational theory of cognition. *Cognitive Science*, *34*, 453–488.

Polya, G. (1957). *How to solve it: A new aspect of mathematical method* (2nd ed.). Garden City, NY: Doubleday Anchor.

Pour-El, M. B. (1974). Abstract computability and its relation to the general purpose analog computer (some connections between logic, differential equations and analog computers). *Transactions of the American Mathematical Society*, *199*, 1–28.

Powell, C. S. (2006, April 2). Welcome to the machine. *New York Times Book Review*, 19. http://www.nytimes.com/2006/04/02/books/review/02powell.html.

Putnam, H. (1965). Trial and error predicates and the solution to a problem of Mostowski. *Journal of Symbolic Logic*, *30*(1), 49–57.

Putnam, H. (1988). *Representation and Reality*. Cambridge, MA: MIT Press.

Randell, B. (1994). The origins of computer programming. *IEEE Annals of the History of Computing*, *16*(4), 6–14.

Rapaport, W. J. (1988). Syntactic semantics: Foundations of computational natural-language understanding. In J. H. Fetzer (Ed.), *Aspects of artificial intelligence* (pp. 81–131). Dordrecht: Kluwer. http://www.cse.buffalo.edu/~rapaport/Papers/synsem.pdf; reprinted with numerous errors in Eric Dietrich (Ed.). (1994). *Thinking machines and virtual persons: Essays on the intentionality of machines* (San Diego: Academic Press): 225–273.

Rapaport, W. J. (1998). How minds can be computational systems. *Journal of Experimental and Theoretical Artificial Intelligence*, *10*, 403–419.

Rapaport, W. J. (1999). Implementation is semantic interpretation. *The Monist*, *82*, 109–130.

Rapaport, W. J. (2005a). Implemention is semantic interpretation: Further thoughts. *Journal of Experimental and Theoretical Artificial Intelligence*, *17*(4), 385–417.

Rapaport, W. J. (2005b). Philosophy of computer science: An introductory course. *Teaching Philosophy*, *28*(4), 319–341.

Rapaport, W. J. (2007). Searle on brains as computers. *American Philosophical Association Newsletter on Philosophy and Computers*, *6*(2), 4–9.

Rapaport, W. J. (2012). Semiotic systems, computers, and the mind: How cognition could be computing. *International Journal of Signs and Semiotic Systems 2*(1), 32–71. Revised version published as Rapaport (2018b).

Rapaport, W. J. (2017a). On the relation of computing to the world. In T. M. Powers (Ed.), *Philosophy and computing: Essays in epistemology, philosophy of mind, logic, and ethics* (pp. 29–64). Cham: Springer. Paper based on 2015 IACAP Covey Award talk; preprint at http://www.cse.buffalo.edu/∼rapaport/Papers/rapaport4IACAP.pdf.

Rapaport, W. J. (2017b). Semantics as syntax. *American Philosophical Association Newsletter on Philosophy and Computers*, *17*(1), 2–11.

Rapaport, W. J. (2017c). What is computer science? *American Philosophical Association Newsletter on Philosophy and Computers*, *16*(2), 2–22. Paper based on 2017 APA Barwise Award talk. http://c.ymcdn.com/sites/www.apaonline.org/resource/collection/EADE8D52-8D02-4136-9A2A-729368501E43/ComputersV16n2.pdf.

Rapaport, W. J. (2018a). *Philosophy of computer science*. Current draft in progress at http://www.cse.buffalo.edu/∼rapaport/Papers/phics.pdf.

Rapaport, W. J. (2018b). Syntactic semantics and the proper treatment of computationalism. In M. Danesi (Ed.), *Empirical research on semiotics and visual rhetoric* (pp. 128–176). Hershey, PA: IGI Global.

Rescorla, M. (2012). How to integrate representation into computational modeling, and why we should. *Journal of Cognitive Science (South Korea)*, *13*(1), 1–38.

Rescorla, M. (2015). The computational theory of mind. In E. N. Zalta (Ed.), *The Stanford encyclopedia of philosophy* (Winter 2015 ed.). http://plato.stanford.edu/archives/win2015/entries/computational-mind/.

Richards, W. (1988). *Natural computation*. Cambridge, MA: MIT Press.

Robinson, J. A. (1994). Logic, computers, Turing, and von Neumann. In K. Furukawa, D. Michie, & S. Muggleton (Eds.), *Machine Intelligence 13: Machine intelligence and inductive learning* (pp. 1–35). Oxford: Clarendon Press.

Romanycia, M. H., & Pelletier, F. J. (1985). What is a heuristic? *Computational Intelligence*, *1*(2), 47–58.

Rubinoff, M. (1953). Analogue vs. digital computers—a comparison. *Proceedings of the IRE*, *41*(10), 1254–1262.

Sackur, J., & Dehaene, S. (2009). The cognitive architecture for chaining of two mental operations. *Cognition*, *111*, 187–211.

Samuel, A. L. (1953). Computing bit by bit, or digital computers made easy. *Proceedings of the IRE*, *41*(10), 1223–1230.

Scheutz, M. (2012). What it is not to implement a computation: A critical analysis of Chalmers' notion of implementation. *Journal of Cognitive Science (South Korea)*, *13*(1), 75–106.

Schmidhuber, J. (2002). *Zuse's thesis: The universe is a computer*. http://www.idsia.ch/∼juergen/digitalphysics.html.

Schmidhuber, J. (2006). The computational universe. *American Scientist*, *94*, 364ff.

Schulman, A. N. (2009). Why minds are not like computers. *The New Atlantis*, *23*, 46–68.

Searle, J. R. (1990). Is the brain a digital computer? *Proceedings and Addresses of the American Philosophical Association*, *64*(3), 21–37. Reprinted in slightly revised form as Searle (1992, Ch. 9).

Searle, J. R. (1992). *The rediscovery of the mind*. Cambridge, MA: MIT Press.

Shagrir, O. (1999). What is computer science about? *The Monist*, *82*(1), 131–149.

Shagrir, O. (2012). Can a brain possess two minds? *Journal of Cognitive Science (South Korea)*, *13*(2), 145–165.

Shapiro, E., & Benenson, Y. (2006). Bringing DNA computers to life. *Scientific American*, *294*(5), 44–51.

Shapiro, S. (2009). We hold these truths to be self-evident: But what do we mean by that? *Review of Symbolic Logic*, *2*(1), 175–207.

Shapiro, S. C. (2001). Computer science: The study of procedures. Technical report, Department of Computer Science and Engineering, University at Buffalo, Buffalo, NY. http://www.cse.buffalo.edu/∼shapiro/Papers/whatiscs.pdf; see also http://www.cse.buffalo.edu/∼shapiro/Courses/CSE115/notes2.html.

Shepherdson, J., & Sturgis, H. (1963). Computability of recursive functions. *Journal of the ACM*, *10*(2), 217–255.

Simon, H. A. (1996). Computational theories of cognition. In W. O'Donohue & R. F. Kitchener (Eds.), *The philosophy of psychology* (pp. 160–172). London: Sage.

Skow, B. (2007). Are shapes intrinsic? *Philosophical Studies*, *133*, 111–130.

Squires, R. (1970). On one's mind. *Philosophical Quarterly*, *20*(81), 347–356.

Sternberg, R. J. (1990). *Metaphors of mind: Conceptions of the nature of intelligence*. New York: Cambridge University Press.
Stewart, I. (1994). A subway named Turing. *Scientific American*, *104*, 106–107.
Stoll, C. (2006). When slide rules ruled. *Scientific American*, *294*(5), 80–87.
Suber, P. (1997). *Formal systems and machines: An isomorphism*. http://www.earlham.edu/~peters/courses/logsys/machines.htm.
Thomason, R. H. (2003). Dynamic contextual intensional logic: Logical foundations and an application. In P. Blackburn (Ed.), *CONTEXT 2003: Lecture notes in artificial intelligence* (Vol. 2680, pp. 328–341). Berlin: Springer.
Turing, A. M. (1936). On computable numbers, with an application to the *Entscheidungsproblem*. *Proceedings of the London Mathematical Society*, *42*, 230–265.
Turing, A. M. (1939). Systems of logic based on ordinals. *Proceedings of the London Mathematical Society*, *S2-45*(1), 161–228. Reprinted with commentary in Copeland (2004, Ch. 3).
Turing, A. M. (1947). Lecture to the London Mathematical Society on 20 February 1947. In B. J. Copeland (Ed.), *The essential Turing* (pp. 378–394). Oxford: Oxford University Press (2004). Editorial commentary on pp. 362–377. https://www.academia.edu/34875977/Alan_Turing_LECTURE_TO_THE_LONDON_MATHEMATICAL_SOCIETY_1947_.
Turing, A. M. (1951, 1996). Intelligent machinery, a heretical theory. *Philosophia Mathematica*, *4*(3), 256–260. http://viola.informatik.uni-bremen.de/typo/fileadmin/media/lernen/Turing-_Intelligent_Machinery.pdf; typescripts and further bibliographic information at http://www.turing.org.uk/sources/biblio1.html.
US National Library of Medicine. (2015). *Dream anatomy*. https://www.nlm.nih.gov/dreamanatomy/da_g_IV-A-02.html.
Vardi, M. Y. (2013). Who begat computing? *Communications of the ACM*, *56*(1), 5.
von Neumann, J. (1945 (1993)). First draft report on the EDVAC. *IEEE Annals of the History of Computing*, *15*(4), 27–75
Wang, H. (1957). A variant to Turing's theory of computing machines. *Journal of the ACM*, *4*(1), 63–92.
Weatherson, B., & Marshall, D. (2018). Intrinsic vs. extrinsic properties. In E. N. Zalta (Ed.), *The Stanford encyclopedia of philosophy* (Spring 2018 ed.). Metaphysics Research Lab, Stanford University. https://plato.stanford.edu/archives/spr2018/entries/intrinsic-extrinsic/.
Wegner, P. (1997). Why interaction is more powerful than algorithms. *Communications of the ACM*, *40*(5), 80–91.
Weinberg, S. (2002). Is the universe a computer? *New York Review of Books*, *49*(16), 43–47.
Weizenbaum, J. (1976). *Computer power and human reason*. New York: W.H. Freeman.
Wheeler, D. L. (1997, October 3). An ancient feud: Who invented the computer? *Chronicle of Higher Education*, B2.
Wolfram, S. (2002a). Introduction to *A new kind of science*. http://www.stephenwolfram.com/publications/introduction-to-a-new-kind-of-science/.
Wolfram, S. (2002b). *A new kind of science*. Wolfram Media. http://www.wolframscience.com/nks/.
Zobrist, A. L. (2000). Computer games: Traditional. In A. Ralston, E. D. Reilly, & D. Hemmendinger (Eds.), *Encyclopedia of Computer Science* (4th ed.). New York: Grove's Dictionaries.
Zylberberg, A., Dehaene, S., Roelfsema, P. R., & Sigman, M. (2011). The human Turing machine: A neural framework for mental programs. *Trends in Cognitive Science*, *15*(7), 293–300.