

## **Chapter Nine**

### **Computationalism**

*Gualtiero Piccinini*

**This is a preprint of an article whose final and definitive form will be published in the *Oxford Handbook of Philosophy and Cognitive Science*. Please refer to the published version.**

Computationalism is the view that cognitive capacities have a computational explanation or, somewhat more strongly, that cognition is (a kind of) computation. For simplicity, I will use these two formulations interchangeably. Most cognitive scientists endorse some version of computationalism and pursue computational explanations as their research program. Thus, when cognitive scientists propose an explanation of a cognitive capacity, the explanation typically involves computations that result in the cognitive capacity.

Computationalism is controversial but resilient to criticism. To understand why mainstream cognitive scientists endorse computationalism and what, if any, alternatives there might be, we need to understand what computationalism says. That, in turn, requires an account of computation.

#### **1. A Substantive Empirical Hypothesis**

Computationalism is usually introduced as an empirical hypothesis that can be disconfirmed. Whether computationalism has empirical bite depends on how we

construe the notion of computation: the more inclusive a notion of computation, the weaker the version of computationalism formulated in its terms.

At one end of the continuum, some notions of computation are so loose that they encompass virtually everything. For instance, if computation is construed as the production of outputs from inputs and if any state of a system qualifies as an input (or output), then every process is a computation. Sometimes, computation is construed as information processing, which is somewhat more stringent, yet the resulting version of computationalism is quite weak. There is little doubt that organisms gather and process information about their environment (more on this below).

Processing information is surely an important aspect of cognition. Thus, if computation is information processing, then cognition involves computation. But this doesn't tell us much about how cognition works. In addition, the notions of information and computation in their most important uses are conceptually distinct, have different histories, are associated with different mathematical theories, and have different roles to play in a theory of cognition. It's best to keep them separate (Piccinini and Scarantino 2010).

Computationalism becomes most interesting when it has explanatory power. The most relevant and explanatory notion of computation is that associated with digital computers. Computers perform impressive feats: they solve mathematical problems, play difficult games, prove logical theorems, etc. Perhaps cognitive systems work like computers. Historically, this analogy between computers and cognitive systems is the main motivation behind computationalism. The resulting form of computationalism is a strong hypothesis—one that should be open to empirical testing. To understand it

further, it will help to briefly outline three research traditions associated with computationalism and how they originated.

## **2. Three Research Traditions: Classicism, Connectionism, and Computational Neuroscience**

The view that thinking has something to do with computation may be found in the works of some modern materialists, such as Thomas Hobbes (Boden 2006, p. 79). But computationalism properly so-called could not begin in earnest until a number of logicians (most notably Alonzo Church, Kurt Gödel, Stephen Kleene, Emil Post, and especially Alan Turing) laid the foundations for the mathematical theory of computation. Turing (1936-7) analyzed computation in terms of what are now called Turing machines—a kind of simple processor operating on an unbounded tape. The tape is divided into squares, which the processor can read and write on. The processor moves along the tape reading and writing on one square at a time depending on what is already on the square as well as on the rules that govern the processor's behavior. The rules state what to write on the tape and where to move next depending on what is on the tape as well as which of finitely many states the processor is in.

Turing argued convincingly that any function that can be computed by following an algorithm (i.e., an unambiguous list of instructions operating on discrete symbols) can be computed by a Turing machine. Church (1936) offered a similar proposal in terms of general recursive functions, and it turns out that a function is general recursive if and only if it can be computed by a Turing machine. Given this extensional equivalence between Turing machines and general recursive functions, the thesis that any algorithmically

computable function is computable by some Turing machine (or equivalently, is general recursive) is now known as the *Church-Turing thesis* (Kleene, 1952, §62, §67).

Turing made two other relevant contributions. First, he showed how to construct *universal* Turing machines. These are Turing machines that can mimic any other Turing machine by encoding the rules that govern the other machine as instructions, storing the instructions on a portion of their tape, and then using the encoded instructions to determine their behavior on the input data. Notice that ordinary digital computers, although they have more complex components than universal Turing machines, are universal in the same sense (up to their memory limitations). That is, digital computers can compute any function computable by a Turing machine until they run out of memory.

Second, Turing showed that the vast majority of functions whose domain is denumerable (e.g., functions of strings of symbols or of natural numbers) are actually *not* computable by Turing machines. These ideas can be put together as follows: assuming the Church-Turing thesis, a universal digital computer can compute any function computable by algorithm, although the sum total of these Turing-computable functions is a tiny subset of all the functions whose domain is denumerable.

Modern computationalism began when Warren McCulloch and Walter Pitts (1943) connected three things: Turing's work on computation, the explanation of cognitive capacities, and the mathematical study of neural networks. Neural networks are sets of connected signal-processing elements ("neurons"). Typically, they have elements that receive inputs from the environment (input elements), elements that yield outputs to the environment (output elements), and elements that communicate only with other elements in the system (hidden elements). Each element receives input signals and

delivers output signals as a function of its input and current state. As a result of their elements' activities and organization, neural networks turn the input received by their input elements into the output produced by their output elements. A neural network may be either a concrete physical system or an abstract mathematical system. An abstract neural network may be used to model another system (such as a network of actual neurons) to some degree of approximation.

The mathematical study of neural networks using biophysical techniques began around the 1930s (Rashevsky, 1938, 1940; Householder and Landahl, 1945). But before McCulloch and Pitts, no one had suggested that neural networks have something to do with computation. McCulloch and Pitts defined networks that operate on sequences of discrete inputs in discrete time, argued that they are a useful idealization of what is found in the nervous system, and concluded that the activity of their networks explains cognitive phenomena. McCulloch and Pitts also pointed out that their networks can perform computations like those of Turing machines. More precisely, McCulloch-Pitts networks are computationally equivalent to Turing machines without tape or finite state automata (Kleene 1956). Notice that modern digital computers are a kind of McCulloch-Pitts neural network (von Neumann, 1945). Digital computers are sets of logic gates—digital signal-processing elements equivalent to McCulloch-Pitts neurons—connected to form a specific architecture.

McCulloch and Pitts's account of cognition contains three important aspects: an analogy between neural processes and digital computations, the use of mathematically defined neural networks as models, and an appeal to neurophysiological evidence to support their neural network models. After McCulloch and Pitts, many others linked

computation and cognition, though they often abandoned one or more aspects of McCulloch and Pitts's theory. Computationalism evolved into three main traditions, each emphasizing a different aspect of McCulloch and Pitts's account.

One tradition, sometimes called *classicism*, emphasizes the analogy between cognitive systems and digital computers while downplaying the relevance of neuroscience to the theory of cognition (Miller, Galanter, and Pribram, 1960; Fodor, 1975; Newell and Simon, 1976; Pylyshyn, 1984; Newell, 1990; Pinker, 1997; Gallistel and King, 2009). When researchers in this tradition offer computational models of a cognitive capacity, the models take the form of computer programs for producing the capacity in question. One strength of the classicist tradition lies in programming computers to exhibit higher cognitive capacities such as problem solving, language processing, and language-based inference.

A second tradition, most closely associated with the term *connectionism* (although this label can be misleading, see Section 7 below), downplays the analogy between cognitive systems and digital computers in favor of computational explanations of cognition that are "neurally inspired" (Rosenblatt, 1958; Feldman and Ballard, 1982; Rumelhart and McClelland, 1986; Bechtel and Abrahamsen, 2002). When researchers in this tradition offer computational models of a cognitive capacity, the models take the form of neural networks for producing the capacity in question. Such models are primarily constrained by *psychological* data, as opposed to *neurophysiological* and *neuroanatomical* data. One strength of the connectionist tradition lies in designing artificial neural networks that exhibit cognitive capacities such as perception, motor control, learning, and implicit memory.

A third tradition is most closely associated with the term *computational neuroscience*, which in turn is one aspect of theoretical neuroscience. Computational neuroscience downplays the analogy between cognitive systems and digital computers even more than the connectionist tradition. Neurocomputational models aim to describe actual neural systems such as (parts of) the hippocampus, cerebellum, or cortex, and are constrained by *neurophysiological* and *neuroanatomical* data in addition to psychological data (Schwartz, 1990; Churchland and Sejnowski, 1992; O'Reilly and Munakata, 2000; Dayan and Abbott, 2001; Eliasmith and Anderson, 2003; Ermentrout and Terman 2010). It turns out that McCulloch-Pitts networks and many of their “connectionist” descendents are relatively unfaithful to the details of neural activity, whereas other types of neural networks are more biologically realistic. These include the following, in order of decreasing biological detail and increasing computational tractability: conductance based models, which go back to Hodgkin and Huxley’s (1952) seminal analysis of the action potential based on conductance changes; networks of integrate-and-fire neurons, which fire simply when the input current reaches a certain threshold (Lapicque, 1907/2007; Caianiello, 1961; Stein, 1965; Knight 1972); and firing rate models, in which there are no individual action potentials—instead, the continuous output of each network unit represents the firing rate of a neuron or neuronal population (Wilson and Cowan 1972). Computational neuroscience offers models of how real neural systems may exhibit cognitive capacities, especially perception, motor control, learning, and implicit memory.

Although the three traditions just outlined are in competition with one another to some extent (more on this in Section 7), there is also some fuzziness at their borders. Some cognitive scientists propose hybrid theories, which combine explanatory resources

drawn from both the classicist and the connectionist traditions (e.g., Anderson, 2007). In addition, biological realism comes in degrees, so there is no sharp divide between connectionist and neurocomputational models.

### **3. Three Accounts of Computation: Causal, Semantic, and Mechanistic**

To fully understand computationalism—the view that cognition is computation—and begin to sort out the disagreements between different computational theories of cognition on one hand and anti-computationalist theories on the other, we need to understand what concrete computation is. Philosophers have offered three main (families of) accounts. (For a more complete survey, see Section 2 of Piccinini, 2010a.)

#### *3.1 The Causal Account*

According to the causal account, a physical system  $S$  performs computation  $C$  just in case (i) there is a mapping from the states ascribed to  $S$  by a physical description to the states defined by computational description  $C$ , such that (ii) the state transitions between the physical states mirror the state transitions between the computational states. Clause (ii) requires that for any computational state transition of the form  $s_1 \rightarrow s_2$  (specified by the computational description  $C$ ), if the system is in the physical state that maps onto  $s_1$ , its physical state *causes* it to go into the physical state that maps onto  $s_2$  (Chrisley, 1995; Chalmers, 1994, 1996; Scheutz, 1999, 2001; see also Klein, 2008 for a similar account built on the notion of disposition rather than cause).

To this causal constraint on acceptable mappings, David Chalmers (1994, 1996) adds a further restriction: a genuine physical implementation of a computational system

must divide into separate physical components, each of which maps onto the components specified by the computational formalism. As Godfrey-Smith (2009, p. 293) notes, this combination of a causal and a localizational constraint goes in the direction of mechanistic explanation (Machamer, Darden, and Craver, 2000). An account of computation that is explicitly based on mechanistic explanation will be discussed below. For now, the causal account simpliciter requires only that the mappings between computational and physical descriptions be such that the causal relations between the physical states are isomorphic to the relations between state transitions specified by the computational description. Thus, according to the causal account, computation is the causal structure of a physical process.

It is important to note that under the causal account, there are mappings between any physical system and at least some computational descriptions. Thus, according to the causal account, everything performs at least some computations (Chalmers, 1996, p. 331; Scheutz, 1999, p. 191). This (limited) *pancomputationalism* strikes some as overly inclusive.<sup>i</sup> By entailing pancomputationalism, the causal account trivializes the claim that a system is computational. For according to pancomputationalism, digital computers perform computations in the same sense in which rocks, hurricanes, and planetary systems do. This does an injustice to computer science—in computer science, only relatively few systems count as performing computations and it takes a lot of difficult technical work to design and build systems that perform computations reliably. Or consider computationalism, which was introduced to shed new and explanatory light on cognition. If every physical process is a computation, computationalism seems to lose much of its explanatory force (Piccinini, 2007a).

Another objection to pancomputationalism begins with the observation that any moderately complex system satisfies indefinitely many objective computational descriptions (Piccinini, 2010b). This may be seen by considering computational modeling. A computational model of a system may be pitched at different levels of granularity. For example, consider cellular automata models of the dynamics of a galaxy or a brain. The dynamics of a galaxy or a brain may be described using an indefinite number of cellular automata—using different state transition rules, different time steps, or cells that represent spatial regions of different sizes. Furthermore, an indefinite number of formalisms different from cellular automata, such as Turing machines, can be used to compute the same functions computed by cellular automata. It appears that pancomputationalists are committed to the galaxy or the brain performing all these computations at once. But that does not appear to be the sense in which computers (or brains) perform computations.

In the face of these objections, supporters of the causal account of computation are likely to maintain that the explanatory force of computational explanations does not come merely from the claim that a system is computational. Rather, explanatory force comes from the specific computations that a system is said to perform. Thus, a rock and a digital computer perform computations in the same sense. But they perform radically different computations, and it is the difference between their computations that explains the difference between the rock and the digital computer. As to the objection that there are still too many computations performed by each system, pancomputationalists have two main options: either to bite the bullet and accept that every system implements indefinitely many computations, or to find a way to single out, among the many

computational descriptions satisfied by each system, the one that is ontologically privileged—the one that captures the computation performed by the system.

Those who are unsatisfied by these replies and wish to avoid pancomputationalism need accounts of computation that are less inclusive than the causal account. Such accounts may be found by adding restrictions to the causal account.

### *3.2 The Semantic Account*

In our everyday life, we usually employ computations to process meaningful symbols, in order to extract information from them. The semantic account of computation turns this practice into a metaphysical doctrine: computation is the processing of representations—or at least, the processing of appropriate representations in appropriate ways (Fodor, 1975; Cummins, 1983; Pylyshyn, 1984; Churchland and Sejnowski, 1992; Shagrir, 2006; Sprevak, 2010). Opinions as to which representational manipulations constitute computations vary. What all versions of the semantic account have in common is that according to them, there is “no computation without representation” (Fodor, 1981, p. 180).

The semantic account may be formulated (and is usually understood) as a restricted causal account. In addition to the causal account’s requirement that a computational description mirror the causal structure of a physical system, the semantic account adds a semantic requirement. Only physical states that qualify as representations may be mapped onto computational descriptions, thereby qualifying as computational states. If a state is not representational, it is not computational either.

The semantic account of computation is closely related to the common view that computation is information processing. This idea is less clear than it may seem, because there are several notions of information. The connection between information processing and computation is different depending on which notion of information is at stake. What follows is a brief disambiguation of the view that computation is information processing based on four important notions of information.

1. Information in the sense of thermodynamics is closely related to thermodynamic entropy. Entropy is a property of every physical system. Thermodynamic entropy is, roughly, a measure of an observer's uncertainty about the microscopic state of a system after she considers the observable macroscopic properties of the system. The study of the thermodynamics of computation is a lively field with many implications in the foundations of physics (Leff and Rex, 2003). In this thermodynamic sense of 'information,' any difference between two distinguishable states of a system may be said to carry information. Computation may well be said to be information processing in this sense, but this has little to do with semantics properly so-called.

2. Information in the sense of communication theory is a measure of the average likelihood that a given message is transmitted between a source and a receiver (Shannon and Weaver, 1949). This has little to do with semantics too.

3. Information in one semantic sense is approximately the same as "natural meaning" (Grice, 1957). A signal carries information in this sense just in case it reliably correlates with a source (Dretske, 1981). The view that computation is

information processing in this sense is *prima facie* implausible, because many computations—such as arithmetical calculations carried out on digital computers—do not seem to carry any natural meaning. Nevertheless, this notion of semantic information is relevant here because it has been used by some theorists to ground an account of representation (Dretske, 1981; Fodor, 2008).

4. Information in another semantic sense is just ordinary semantic content or “non-natural meaning” (Grice, 1957). The view that computation is information processing in this sense is similar to a generic semantic account of computation.

The semantic account is popular in the philosophy of mind and in cognitive science because it appears to fit its specific needs better than a purely causal account. Since minds and computers are generally assumed to manipulate (the right kind of) representations, they turn out to compute. Since most other systems are generally assumed not to manipulate (the relevant kind of) representations, they do not compute. Thus, the semantic account appears to avoid pancomputationalism and to accommodate some common intuitions about what does and does not count as a computing system. It keeps minds and computers *in* while leaving most everything else *out*, thereby vindicating computationalism as a strong and nontrivial theory.

The semantic account faces its share of problems too. For starters, representation does not seem to be presupposed by the notion of computation employed in at least some areas of cognitive science as well as computability theory and computer science—the very sciences that gave rise to the notion of computation at the origin of the computational theory of cognition (Piccinini, 2008a). If this is correct, the semantic

account may not even be adequate to the needs of philosophers of mind—at least those philosophers of mind who wish to make sense of the analogy between minds and the systems designed and studied by computer scientists and computability theorists. Another criticism of the semantic account is that specifying the kind of representation and representational manipulation that is relevant to computation seems to require a non-semantic way of individuating computations (Piccinini, 2004). These concerns motivate efforts to account for concrete computation in non-semantic terms.

### *3.3 The Mechanistic Account*

An implicit appeal some aspects of mechanisms may be found in many accounts of computation, usually in combination with an appeal to causal or semantic properties (Chalmers, 1996; Cummins, 1983; Egan, 1995; Fodor, 1980; Glymour, 1991; Horst, 1999; Newell, 1980; Pylyshyn, 1984; Shagrir, 2001; and Stich, 1983). Nevertheless, the received view is that computational explanations and mechanistic explanations are distinct and belong at different “levels” (Marr 1982, Rusanen and Lappi 2007). By contrast, this section introduces an explicitly mechanistic account of computation, which does not rely on semantics (Piccinini, 2007b; Piccinini and Scarantino, 2010, Section 3). According to this account, computational explanation is a species of mechanistic explanation; concrete computing systems are functionally organized mechanisms of a special kind—mechanisms that perform concrete computations.

Like the semantic account, the mechanistic account may be understood as a restricted causal account. In addition to the causal account’s requirement that a computational description mirror the causal structure of a physical system, the

mechanistic account adds a requirement about the functional organization of the system. Only physical states that have a specific functional significance within a specific type of mechanism may be mapped onto computational descriptions, thereby qualifying as computational states. If a state lacks the appropriate functional significance, it is not a computational state.

A functional mechanism is a system of organized components, each of which has functions to perform (cf. Craver, 2007; Wimsatt, 2002). When appropriate components and their functions are appropriately organized and functioning properly, their combined activities constitute the capacities of the mechanism. Conversely, when we look for an explanation of the capacities of a mechanism, we decompose the mechanism into its components and look for their functions and organization. The result is a mechanistic explanation of the mechanism's capacities.

This notion of mechanism is familiar to biologists and engineers. For example, biologists explain physiological capacities (digestion, respiration, etc.) in terms of the functions performed by systems of organized components (the digestive system, the respiratory system, etc.).

According to the mechanistic account, a computation in the generic sense is the processing of vehicles according to rules that are sensitive to certain vehicle properties, and specifically, to differences between different portions of the vehicles. The processing is performed by a functional mechanism, that is, a mechanism whose components are functionally organized to perform the computation. Thus, if the mechanism malfunctions, a miscomputation occurs.

When we define concrete computations and the vehicles that they manipulate, we need not consider all of their specific physical properties. We may consider only the properties that are relevant to the computation, according to the rules that define the computation. A physical system can be described at different levels of abstraction. Since concrete computations and their vehicles are described sufficiently abstractly as to be defined independently of the physical media that implement them, they may be called *medium-independent*.

In other words, a vehicle is medium-independent just in case the rules (i.e., the input-output maps) that define a computation are sensitive only to differences between portions of the vehicles along specific dimensions of variation—they are insensitive to any more concrete physical properties of the vehicles. Put yet another way, the rules are functions of state variables associated with a set of functionally relevant degrees of freedom, which can be implemented differently in different physical media. Thus, a given computation can be implemented in multiple physical media (e.g., mechanical, electro-mechanical, electronic, magnetic, etc.), provided that the media possess a sufficient number of dimensions of variation (or degrees of freedom) that can be appropriately accessed and manipulated and that the components of the mechanism are functionally organized in the appropriate way.

Notice that the mechanistic account avoids pancomputationalism. First, physical systems that are not functional mechanisms are ruled out. Functional mechanisms are complex systems of components that are organized to perform functions. Any system whose components are not organized to perform functions is not a computing system because it is not a functional mechanism. Second, mechanisms that lack the function of

manipulating medium-independent vehicles are ruled out. Finally, medium-independent vehicle manipulators whose manipulations fail to accord with appropriate rules are ruled out. The second and third constraints appeal to special functional properties—manipulating medium-independent vehicles, doing so in accordance with rules defined over the vehicles—that are possessed only by relatively few concrete mechanisms. According to the mechanistic account, those few mechanisms are the genuine computing systems.

Another feature of the mechanistic account is that it makes sense of miscomputation—a notion difficult to make sense of under the causal and semantic accounts. Consider an ordinary computer programmed to compute function  $f$  on input  $i$ . Suppose that the computer malfunctions and produces an output different from  $f(i)$ . According to the causal (semantic) account, the computer just underwent a causal process (a manipulation of representations), which may be given a computational description and hence counts as computing some function  $g(i)$ , where  $g \neq f$ . By contrast, according to the mechanistic account, the computer simply failed to compute, or at least failed to complete its computation. Given the importance of avoiding miscomputations in the design and use of computers, the ability of the mechanistic account to make sense of miscomputation gives it a further advantage over rival accounts.

The most important advantage of the mechanistic account over other accounts is that it distinguishes and characterizes precisely many different kinds of computing systems based on their specific mechanistic properties. Given its advantages, from now on I will presuppose the mechanistic account. I will now describe some important kinds of computation.

## 4. Kinds of Computation: Digital, Analog, and More

### 4.1 Digital Computation

The best-known kind of computation is digital computation. To a first approximation, digital computation, as I am using this term, is the kind of computation that was analyzed by the classical mathematical theory of computation that began with Alan Turing and other logicians and is now a well-established branch of mathematics (Davis et al., 1994).

Digital computation may be defined both abstractly and concretely. Roughly speaking, abstract digital computation is the manipulation of strings of discrete elements, that is, strings of letters from a finite alphabet. Here we are interested primarily in concrete computation, or physical computation. Letters from a finite alphabet may be physically implemented by what I call *digits*. To a first approximation, concrete digital computation is the processing of sequences of digits according to general rules defined over the digits (Piccinini, 2007b). Let us briefly consider the main ingredients of digital computation.

The atomic vehicles of concrete digital computation are digits, where a digit is a macroscopic state (of a component of the system) whose type can be reliably and unambiguously distinguished by the system from other macroscopic types. To each (macroscopic) digit type, there correspond a large number of possible microscopic states. Artificial digital systems are engineered so as to treat all those microscopic states in the same way—the one way that corresponds to their (macroscopic) digit type. For instance, a system may treat 4 volts plus or minus some noise in the same way (as a ‘0’), whereas it may treat 8 volts plus or minus some noise in a different way (as a ‘1’). To ensure reliable manipulation of digits based on their type, a physical system must manipulate at

most a finite number of digit types. For instance, ordinary computers contain only two types of digit, usually referred to as ‘0’ and ‘1’. Digits need not mean or represent anything, but they can. For instance, numerals represent numbers, while other digits (e.g., ‘|’, ‘\’) do not represent anything in particular.

Digits can be concatenated (i.e., ordered) to form sequences or *strings*. Strings of digits are the vehicles of digital computations. A digital computation consists in the processing of strings of digits according to rules. A rule for a digital computation is simply a map from input strings of digits, plus possibly internal states, to output strings of digits. Examples of rules that may figure in a digital computation include addition, multiplication, identity, and sorting.

Digits are unambiguously distinguishable by the processing mechanism under normal operating conditions. Strings of digits are ordered sets of digits, i.e., digits such that the system can distinguish different members of the set depending on where they lie along the string. The rules defining digital computations are, in turn, defined in terms of strings of digits and internal states of the system, which are simply states that the system can distinguish from one another. No further physical properties of a physical medium are relevant to whether its states implement digital computations. Thus, digital computations can be implemented by any physical medium with the right degrees of freedom.

To summarize, a physical system is a digital computing system just in case it is a system that manipulates input strings of digits, depending on the digits’ type and their location on the string, in accordance with a rule defined over the strings (and possibly certain internal states of the system).

The notion of digital computation here defined is quite general. It should not be confused with three other commonly invoked but more restrictive notions of computation: classical computation in the sense of Fodor and Pylyshyn (1988), algorithmic computation, and computation of Turing-computable functions (see Figure 1).

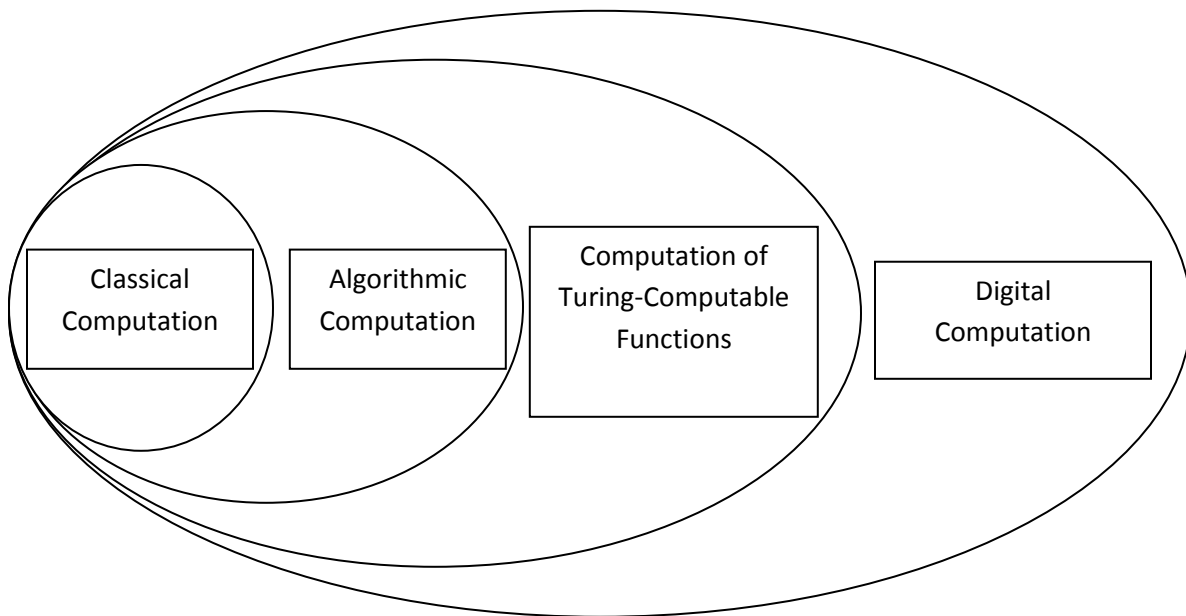


Figure 1. Types of digital computation and their relations of class inclusion.

Let us begin with the most restrictive notion of the three: classical computation. A classical computation is a digital computation that has two additional features. First, it manipulates a special kind of digital vehicle: sentence-like structures. Second, it is algorithmic: it proceeds in accordance with effective, step-by-step procedures that manipulate strings of digits and produce a result within finitely many steps. Thus, a classical computation is a digital, algorithmic computation whose algorithms are sensitive to the combinatorial syntax of the strings (Fodor and Pylyshyn 1988).

A classical computation is algorithmic, but the notion of algorithmic computation—digital computation that follows an algorithm—is more inclusive, because it does not require that the vehicles being manipulated be sentence-like.

Any algorithmic computation, in turn, can be performed by some Turing machine (i.e., the computed function is Turing-computable). This is a version of the Church-Turing thesis. But the computation of Turing-computable functions need not be carried out by following an algorithm. For instance, many neural networks compute Turing-computable functions (their inputs and outputs are strings of digits, and the input-output map is Turing-computable), but such networks need not have a level of functional organization at which they follow the steps of an algorithm for computing their functions; there may be no functional level at which their internal states and state transitions are discrete.

Here is another way to draw the distinction between algorithmic digital computation and digital computation more generally. Algorithmic digital computation is *fully* digital—digital every step of the way. Fully digital systems, such as McCulloch-Pitts networks (including, of course, digital computers), produce digital outputs from digital inputs by means of discrete intermediate steps. Thus, the computations of fully digital computing systems can be characterized as the (step-by-step) algorithmic manipulations of strings of digits. By contrast, digital computations more generally are only *input/output (I/O) digital*. I/O digital systems, including many neural networks, produce digital outputs from digital inputs by means of irreducibly continuous intermediate processes. Thus, the computations of merely I/O digital computing systems cannot be characterized as step-by-step algorithmic manipulations of strings of digits.<sup>ii</sup>

Finally, the computation of a Turing-computable function is a digital computation, because Turing-computable functions are by definition functions of a denumerable domain—a domain whose elements may be counted—and the arguments and values of such functions are, or may be represented by, strings of digits. But it is equally possible to define functions of strings of digits that are not Turing-computable, and to mathematically define processes that compute such functions. Some authors have speculated that some functions over strings of digits that are not Turing-computable may be computable by some physical systems (Penrose, 1994; Copeland, 2000). According to the present usage, any such computations still count as digital computations. It may well be that only the Turing-computable functions are computable by physical systems—whether this is the case is an empirical question that does not affect our discussion. Furthermore, the computation of Turing-uncomputable functions is unlikely to be relevant to the study of cognition (more on this in Section 6.6). Be that as it may—I will continue to talk about digital computation in general.

Many other distinctions may be drawn within digital computation, such as hardwired vs. programmable, special purpose vs. general purpose, and serial vs. parallel (cf. Piccinini, 2008b). These distinctions play an important role in debates about the computational architecture of cognitive systems, but I lack the space to discuss them here.

In summary, digital computation includes many types of neural network computations, including processes that follow ordinary algorithms, such as the computations performed by standard digital computers. Since digital computation is the

notion that inspired the computational theory of cognition, it is the most relevant notion for present purposes.

#### *4.2 Analog Computation*

Analog computation is often contrasted with digital computation, but analog computation is a vaguer and more slippery concept. The clearest notion of analog computation is that of Pour-El (1974). Roughly, abstract analog computers are systems that manipulate continuous variables—variables that can vary continuously over time and take any real values within certain intervals—to solve certain systems of differential equations.

Analog computers can be physically implemented, and physically implemented continuous variables are different kinds of vehicles than strings of digits. While a digital computing system can always unambiguously distinguish digits and their types from one another, a concrete analog computing system cannot do the same with the exact values of (physically implemented) continuous variables. This is because continuous variables can take any real values but there is a lower bound to the sensitivity of any physical system. Thus, it is always possible that the difference between two portions of a continuous variable is small enough to go undetected by the system. From this it follows that the vehicles of analog computation are not strings of digits and analog computations (in the present, strict sense) are a different kind of process than digital computations.

Nevertheless, analog computations are only sensitive to the differences between portions of the variables being manipulated, to the degree that they can be distinguished by the system. Any further physical properties of the media implementing the variables

are irrelevant to the computation. Like digital computers, therefore, analog computers operate on medium-independent vehicles.

There are other kinds of computation besides digital and analog computation. One increasingly popular kind, *quantum computation*, is an extension of digital computation in which digits are replaced by quantum states called *qudits* (most commonly, binary qudits, which are known as *qubits*). There is no room here for a complete survey of kinds of computation. Suffice it to conclude that computation in the generic sense includes digital computation, analog computation, quantum computation, and more (Figure 2).

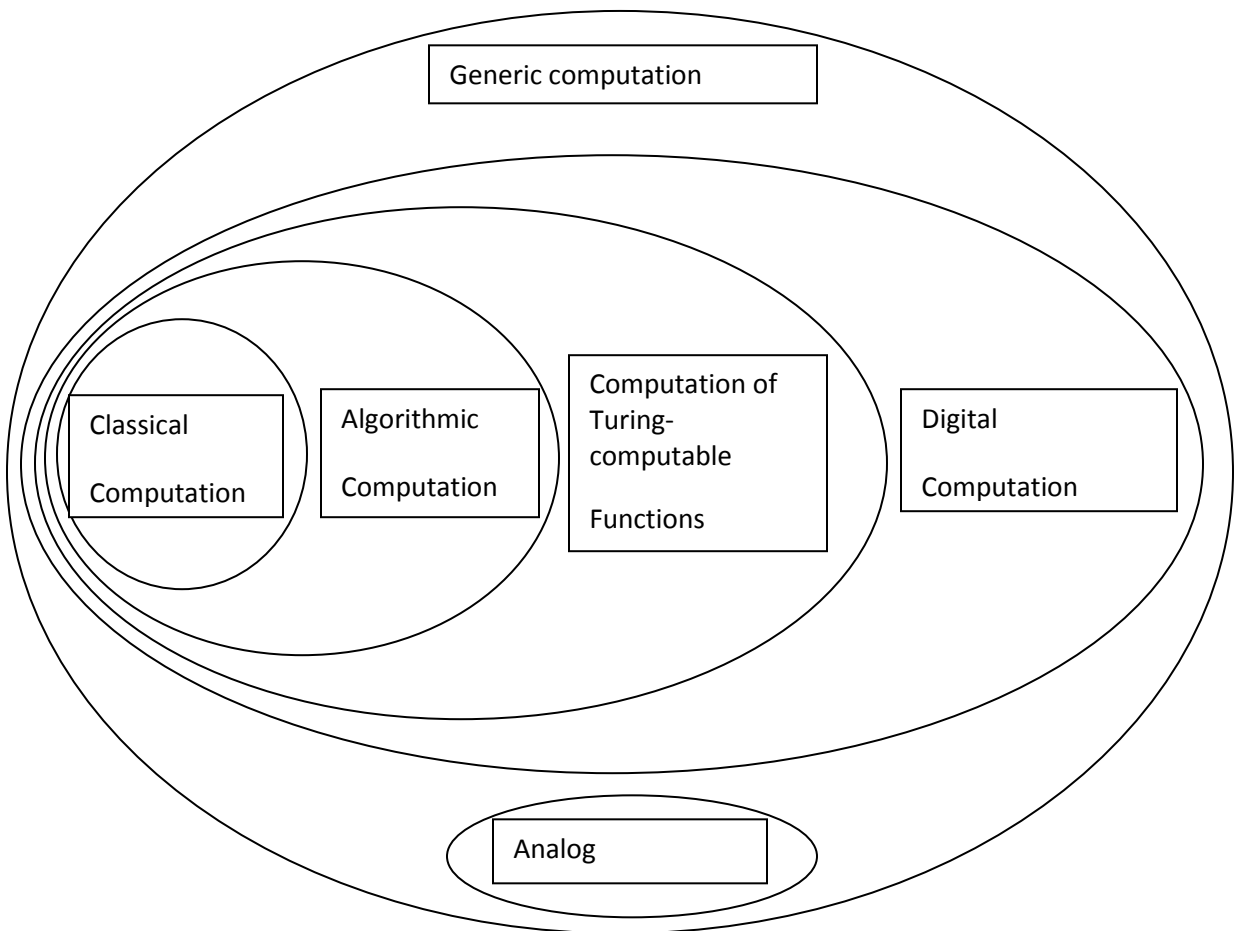


Figure 2. Types of computation and their relations of class inclusion.

## 5. Computation and Cognition: Digital, Analog, or a Third Kind?

Computationalism takes different forms depending on which type of computation it invokes to explain cognitive capacities. The weakest form of computationalism is *generic computationalism*, which simply says that cognition is a form of computation in the generic sense. Stronger versions of computationalism are *analog computationalism* and *digital computationalism*, according to which cognition is, respectively, (a kind of) analog or digital computation. Digital computationalism comes in more specialized versions that invoke different kinds of digital computation. A historically influential version of digital computationalism is *classical computationalism*, according to which cognition is (a kind of) classical computation. An important aspect of debates over the nature of cognition focuses on which type of computation cognition is. Let's begin with two reasons in favor of generic computationalism. Later we will mention some arguments for more specific versions of computationalism.

The vehicles of neural processes are medium-independent, so their manipulation constitutes a computation in the generic sense. As Edgar Adrian (1928; cf. Garson 2003) first showed, the vehicles transmitted and manipulated by neural processes are neuronal spikes (action potentials) and the functionally relevant aspects of neural processes depend on medium-independent aspects of the spikes—primarily, spike *rates*.<sup>iii</sup> The functionally relevant aspects of spikes may be implemented either by neural tissue or by some other physical medium, such as a silicon-based circuit. Thus, spike trains—sequences of spikes produced by neurons in real time—appear to be medium-independent vehicles, in which case they qualify as proper vehicles for computations in the generic sense.

Assuming that brains process spike trains and that spikes are medium-independent vehicles, it follows that brains perform computations in the generic sense.

A second reason for generic computationalism begins with the observation that cognition involves the processing of information, in at least three important senses. First, cognition requires processing stochastic signals; this is information in Shannon's non-semantic sense (Shannon and Weaver, 1949). Second, cognition requires processing signals that causally correlate with their sources; this may be called *natural semantic information* (cf. Dretske, 1981). Third, at least some forms of cognition require processing full-blown representations, i.e., internal states that can represent either correctly or incorrectly; this may be called *non-natural semantic information* (Piccinini and Scarantino, 2010).

For those who identify computation and information processing, the fact that cognition involves information processing is enough to conclude that cognition involves computation. As I pointed out, however, computation and information processing are best kept conceptually distinct, if nothing else because identifying them obliterates important distinctions between different notions of computation and information.

Nevertheless, the fact that cognition involves information processing does entail that cognition involves computation. This is because information (in all senses of the term) is a medium-independent notion. It doesn't take any particular physical medium with any particular physical properties to possess certain statistical properties (Shannon information), causally correlate with a source (natural semantic information), or represent (non-natural semantic information). Thus, processing information requires processing medium-independent vehicles. Recall that computation in the generic sense is the

processing of medium-independent vehicles. Thus, information processing entails computation in the generic sense. (The converse does not hold because medium-independent vehicles need not carry any information; this is another reason to deny that computation is information processing.)

While it's safe to say that cognition involves computation in the generic sense and that nervous systems perform computations in the generic sense, it's much harder to establish that cognition involves a more specific kind of computation.

We've already encountered McCulloch and Pitts's argument that cognition involves digital computation because of the all-or-none nature of the action potential. This suggestion encountered mighty resistance and is now abandoned by most neuroscientists. One reason is that unlike strings of digits properly so called, action potentials do not appear to be organized into discrete time intervals of well defined functional significance.

Shortly after McCulloch and Pitts argued that brains perform digital computations, others countered that neural processes may be more similar to analog computations (Gerard, 1951; see also Rubel, 1985). The evidence for the analog computational theory included neurotransmitters and hormones, which are released by neurons in degrees rather than in all-or-none fashion.

The claim that the brain is an analog computer is ambiguous between two interpretations. On the literal interpretation, 'analog computer' is given Pour-El's (1974) precise meaning. The theory that the brain is an analog computer in this literal sense was never very popular. The primary reason is that although neural signals are a continuous function of time, they are also sequences of all-or-none signals.

On a looser interpretation, ‘analog computer’ refers to a broader class of computing systems. For instance, Churchland and Sejnowski use the term ‘analog’ so that containing continuous variables is sufficient for a system to count as analog: “The input to a neuron is analog (continuous values between 0 and 1)” (1992, p. 51). Under such a usage, even a slide rule counts as an analog computer. Sometimes, the notion of analog computation is simply left undefined, with the result that by ‘analog computer’ one refers to some kind of presumably non-digital but otherwise unspecified computing system.

The looser interpretation of ‘analog computer’—of which Churchland and Sejnowski’s usage is one example—is not uncommon, but I find it misleading because prone to being confused with the literal interpretation of analog computation in the sense of Pour-El (1974).

Not all arguments for specific versions of computationalism are based on neuroscientific evidence. In fact, typical arguments for classical computationalism make no reference to the brain. One argument for classical computationalism begins with the observation that certain cognitive capacities exhibit productivity and systematicity—roughly speaking, they can generate an indefinite number of systematically related structured behaviors such as natural language sentences. The argument assumes that productive and systematic cognition requires the processing of language-like representations on the basis of their syntactic structure—that is, classical computation—and concludes that cognition is a kind of classical computation (Fodor and Pylyshyn 1988, Aizawa 2003). Like other arguments for specific versions of computationalism, this argument has encountered resistance—either from people who deny that cognition

involves the processing of language-like representations or from people who deny that the relevant processing of language-like representations requires a classical computational system.

Another argument for classical computationalism is the argument from cognitive flexibility. Human beings can solve an indefinite number of problems and learn an indefinite range of behaviors. How do they do it? Consider computers—the most flexible of artifacts. Computers can do mathematical calculations, derive logical theorems, play board games, recognize objects, control robots, and even engage in conversations to some degree. They can do this because they can execute different sets of instructions designed for different tasks. In Turing's terms, they approximate universal machines. Perhaps human beings are cognitively flexible because, like computers, they possess a general purpose processing mechanism that executes different instructions for different tasks (Fodor, 1968; Newell, 1990; Samuels, 2010). The argument from cognitive flexibility is one of the most powerful, because there is no well worked-out alternative explanation of cognitive flexibility, at least for high level cognitive skills such as problem solving (and according to some, even for lower level cognitive skills, cf. Gallistel and King, 2009). Of course, the argument from cognitive flexibility is controversial too.

In conclusion, it is very plausible that the neural processes that explain cognitive capacities are computational in the generic sense, but it is difficult to determine which specific kinds of computation—classical, digital but non-classical, analog, etc.—are involved in which cognitive capacities. Whether any particular neural computation is

best regarded as a form of digital computation, analog computation, or something else is a question that we cannot settle here.

## **6. Some Objections and Proposed Alternatives to Computationalism**

Computationalism, we have said, is the view that cognition is a kind of computation. We now appreciate that this may mean a number of different things. The original form of modern computationalism is *digital computationalism*—cognition as (a kind of) digital computation. A weaker thesis is *generic computationalism*—cognition as (a kind of) computation in the generic sense. In evaluating the following objections to computationalism, we should keep in mind whether they impugn only digital computationalism or also generic computationalism, and whether they give rise to an alternative way of studying cognition.

### *6.1 Consciousness*

Some argue that computation is insufficient for phenomenal consciousness (e.g., Block 1978). If computation is insufficient for phenomenal consciousness and cognition involves phenomenal consciousness, then, at the very least, computation is insufficient for cognition.

Whether and in what way cognition involves phenomenal consciousness is controversial. Some philosophers maintain that consciousness is epiphenomenal and that cognition can be explained without involving consciousness. If so, then the objection from consciousness is defused. Even if cognition involves consciousness, many cognitive scientists maintain that consciousness is reducible to computation and

information processing (cf. Baars, Banks, and Newman, 2003), and several philosophers have attempted to explain consciousness in broadly computational terms (e.g., Lycan, 1987; Dennett, 1991; Rey, 2005). Finally, even if cognition involves consciousness and consciousness requires something beyond computation, it may be possible to study cognition independently of consciousness. In any case, the objection from consciousness does not give rise to a non-computational scientific approach to cognition; for all that the objection says, computation may remain an important part of the explanation of cognition.

### *6.2 Intentionality/Understanding*

Some argue that cognition involves intentionality (i.e., that cognitive states are about something), but computation is insufficient for intentionality. Therefore, cognition is not computation. A well known variant of this objection claims that computation is insufficient for “understanding” (Searle, 1980).

One possible response is that cognition does not involve intentionality. According to this response, explaining behavior does not require the postulation of mental representations or similarly intentional notions. Another response is that computation does suffice for intentionality. The semantic view of computation may seem to help here. For if computation presupposes representation in a robust enough sense (“original intentionality”), then computation is automatically intentional. The problem with this second response is that computation per se hardly presupposes the right notion of representation (“original intentionality”).

Most intentional realists prefer a third response: they accept that computation is insufficient for intentionality but maintain that computation is still the process that explains cognitive capacities. According to them, cognition is *intentional* computation. Where does the intentionality come from? Some of them take intentionality as a primitive or give it a non-reductive account, while others offer naturalistic accounts of intentionality. Most of today's computationalists seem to accept that a complete explanation of cognitive phenomena requires not only a computational explanation of behavior, but also an account of intentionality.

### *6.3 Anti-Representationalism*

Some argue that computationalism presupposes that cognition manipulates representations, but cognition does no such thing. Therefore, cognition is not computation (cf. van Gelder, 1995). This objection reverses the previous one, and it's a nonstarter. There are two reasons.

First, computationalism can be formulated without presupposing representationalism, as I have done in this chapter. Second, representations are a staple of mainstream psychology and neuroscience. They are unlikely to be eliminated from our explanations of cognition.

### *6.4 Embodied and Embedded Cognition*

Some argue that cognition is embodied (i.e., coupled with a body) and embedded (i.e., coupled with the environment), while computation is disembodied and unembedded. Therefore, cognition is not computation (cf. Thompson, 2007). This objection takes a

variety of forms depending on what is meant by embodiment and embeddedness. In any case, the objection is based on a false premise.

Computation *can* be disembodied and unembedded but it *need not* be. Computing systems can be coupled with a body, an environment, or both. For any substantive version of the thesis that cognition is embodied or embedded (i.e., any version that does not build anti-computationalism into its definition), computations remain at least part of the cognitive processes. And for those with a taste for extended cognition (i.e., cognitive systems some of whose parts are in the agent's environment), Robert Wilson (1994) has proposed a version of computationalism according to which the computations themselves extend into the environment. Thus, computationalism is fully compatible with the study of embodied and embedded cognition.

### 6.5 *Dynamical Systems*

Some argue that cognition is dynamical, not computational (cf. van Gelder, 1998). They propose explaining cognitive capacities not as the output of computations but rather as the behavior of dynamical systems (Port and van Gelder, 1995; Spivey, 2007).

This argument presupposes a contrast between dynamical systems and computational ones, but that is a false contrast. Dynamical systems are systems that change over time as a function of their current state. Dynamical systems are usefully modeled using systems of differential equations or difference equations, and the study of such equations is called 'dynamical systems theory'. Cognitive systems, of course, are dynamical systems—they change over time as a function of their current state. Thus, some scientists fruitfully employ differential equations to study cognitive systems at

various levels of organization. By the same token, computational systems are dynamical systems too, and they are often studied using differential or difference equations. In fact, mathematical tools derived from dynamical systems theory figure prominently in most types of computational explanations of cognitive capacities. Thus, while there may be opposition between a particular type of dynamics (say, an irreducibly continuous dynamics) and a particular type of computation (say, algorithmic digital computation), in general there is no genuine opposition between the claim that cognition is dynamical and the claim that cognition is computation.

### *6.6 The Mathematical Objection*

As Turing showed, there is a precise limit to the range of theorems that any (fixed) computing machine can prove. But as Turing also pointed out, mathematicians are capable of inventing new methods of proof, so they can prove more theorems than any (fixed) computing machine. Therefore, the objection concludes, the cognitive systems of human mathematicians are not computing machines (Turing, 1950).

Some people have taken the mathematical objection a step further. Rather than claiming merely that cognition involves something beyond computation, they argue that cognition involves hypercomputations—that is, computations more powerful than those carried out by Turing machines (Copeland, 2000; Siegelmann, 2003; Bringsjord and Arkoudas, 2007). Based on our taxonomy (Section 4), this is still a version of digital computationalism—it just says that the computations performed by cognitive systems are more powerful than those of Turing machines. But at any rate, there is no hard evidence that human beings can perform hypercomputations: no one has shown how to solve a

genuinely Turing-uncomputable problem, such as the halting problem,<sup>iv</sup> by using human cognitive faculties. Therefore, for now, all we need to answer is Turing's original mathematical objection.

As Turing pointed out, computing machines that change their programs over time and are allowed to make mistakes are not limited in the way that fixed machines are. They can enlarge the range of methods they can employ and the theorems they can prove. Therefore, the cognitive systems of human mathematicians can still be computing machines, so long as they change over time (in a non-computable way) and can make mistakes. Since this last claim is independently plausible, computationalism escapes the mathematical objection (Piccinini, 2003).

## **7. Classicism, Connectionism, and Computational Neuroscience Revisited**

The distinctions introduced in the preceding sections allow us to shed new light on the longstanding debates between classicism, connectionism, and computational neuroscience.

By the 1970s, McCulloch and Pitts were mostly forgotten. The dominant computational paradigm in cognitive science was classical or "symbolic" AI, aimed at writing computer programs that simulate intelligent behavior without much concern for how brains work (e.g., Newell and Simon, 1976). It was commonly assumed that digital computationalism is committed to classicism, that is, the idea that cognition is the manipulation of language-like representations. Language-like representations have constituent structure whose computational manipulation, according to classicism, explains productive and systematic psychological processes. On this view, cognition

consists of performing computations on sentences with a logico-syntactic structure akin to that of natural languages, but written in the language of thought (Harman, 1973; Fodor, 1975).

It was also assumed that, given digital computationalism, explaining cognition is independent of explaining neural activity, in the sense that figuring out how the brain works tells us little or nothing about how cognition works: neural (or implementational, or mechanistic) explanations and computational explanations are at two different “levels” (Fodor, 1975; Marr, 1982).

During the 1980s, connectionism re-emerged as an influential approach to psychology. Most connectionists deny that explaining cognition requires postulating a language of thought and affirm that a theory of cognition should be at least “inspired” by the way the brain works (Rumelhart and McClelland, 1986).

The resulting debate between classicists and connectionists (e.g., Fodor and Pylyshyn, 1988; Smolensky, 1988) has been somewhat confusing. Different authors employ different notions of computation, which vary in both their degree of precision and their inclusiveness. Specifically, some authors use the term ‘computation’ only for classical computation—i.e., at a minimum, algorithmic digital computation over language-like structures—and conclude that (non-classicist) connectionism falls outside computationalism. By contrast, other authors use a broader notion of computation along the lines of what I called computation in the generic sense, thus including connectionism within computationalism (see Piccinini, 2008c for more details). But even after we factor out differences in notions of computation, further confusions lie in the wings.

Classical computationalism and connectionism are often described as being at odds with one another, because classical computationalism is committed to classical computation (the idea that the vehicles of digital computation are language-like structures) and—it is assumed—to autonomy from neuroscience, two theses flatly denied by many prominent connectionists. But many connectionists also model and explain cognition using neural networks that perform computations defined over strings of digits, so perhaps they should be counted among the digital computationalists (Bechtel & Abrahamsen, 2002; Feldman & Ballard, 1982; Hopfield, 1982; Rumelhart & McClelland, 1986; Smolensky & Legendre, 2006).

Furthermore, both classicists and connectionists tend to ignore computational neuroscientists, who in turn tend to ignore both classicism and connectionism. Computational neuroscientists often operate with their own mathematical tools without committing themselves to a particular notion of computation (O'Reilly and Munakata, 2000; Dayan and Abbott, 2001; Eliasmith and Anderson, 2003; Ermentrout and Terman 2010). To make matters worse, some connectionists and computational neuroscientists reject digital computationalism—they maintain that their neural networks, while explaining behavior, do not perform digital computations (Edelman, 1992; Freeman, 2001; Globus, 1992; Horgan & Tienson, 1996; Perkel, 1990; Spivey, 2007).

In addition, the very origin of digital computationalism calls into question the commitment to autonomy from neuroscience. McCulloch and Pitts initially introduced digital computationalism as a theory of the brain, and some form of computationalism or other is now a working assumption of many neuroscientists.

To clarify this debate, we need two separate distinctions. One is the distinction between digital computationalism (“cognition is digital computation”) and its denial (“cognition is something other than digital computation”). The other is the distinction between classicism (“cognition is algorithmic digital computation over language-like structures”) and neural-network approaches (“cognition is computation—digital or not—by means of neural networks”).

We then have two versions of digital computationalism—the classical one (“cognition is algorithmic digital computation over language-like structures”) and the neural network one (“cognition is digital computation by neural networks”)—standing opposite to the denial of digital computationalism (“cognition is a kind of neural network computation different from digital computation”). This may be enough to accommodate most views in the current debate. But it still doesn’t do justice to the relationship between classicism and neural networks.

A further wrinkle in this debate derives from the ambiguity of the term ‘connectionism’. In its original sense, connectionism says that behavior is explained by the changing ‘connections’ between stimuli and responses, which are biologically mediated by changing connections between neurons (Thorndike, 1932; Hebb, 1949). This original connectionism is related to behaviorist associationism, according to which behavior is explained by the association between stimuli and responses. Associationist connectionism adds a biological mechanism to explain the associations: the mechanism of changing connections between neurons.

But contemporary connectionism is a more general thesis than associationist connectionism. In its most general form, contemporary connectionism, like

computational neuroscience, simply says that cognition is explained (at some level) by neural network activity. But this is a truism—or at least it should be. The brain is the organ of cognition, the cells that perform cognitive functions are (mostly) neurons, and neurons perform their cognitive labor by organizing themselves in networks. In Section 5 we saw some that neural activity is computation at least in the generic sense. And in Section 2 we saw that even digital computers are just one special kind of neural network. So even classicists, whose theory is most closely inspired by digital computers, are committed to connectionism in its general sense.

The relationship between connectionist and neurocomputational approaches on one hand and associationism on the other is more complex than many suppose. We should distinguish between strong and weak associationism. Strong associationism maintains that association is the only legitimate explanatory construct in a theory of cognition (cf. Fodor 1983, p. 27). Weak associationism maintains that association is a legitimate explanatory construct along with others such as the innate structure of neural systems.

To be sure, some connectionists profess strong associationism (e.g., Rosenblatt 1958, p. 387). But that is beside the point, because connectionism per se is consistent with weak associationism or even the complete rejection of associationism. Some connectionist models do not rely on association at all—a prominent example being the work of McCulloch and Pitts (1943). And weak associationism is consistent with many theories of cognition including classicism. A vivid illustration is Alan Turing's early proposal to train associative neural networks to acquire the architectural structure of a universal computing machine (Turing 1948, Copeland and Proudfoot 1996). In Turing's

proposal, association may explain how a network acquires the capacity for universal computation (or an approximation thereof), while the capacity for universal computation may explain any number of other cognitive phenomena.

Although many of today's connectionists and computational neuroscientists emphasize the explanatory role of association, many of them also combine association with other explanatory constructs, as per weak associationism (cf. Smolensky and Legendre 2006, p. 479; Trehub 1991, pp. 243-5; Marcus 2001, pp. xii, 30). What remains to be determined is which neural networks, organized in what way, actually explain cognition and which role association and other explanatory constructs should play in a theory of cognition.

Yet another source of confusion is that classicism, connectionism, and computational neuroscience tend to offer explanations at different mechanistic levels. Specifically, classicists tend to offer explanations in terms of rules and representations, without detailing the neural mechanisms by which the representations are implemented and processed; connectionists tend to offer explanations in terms of highly abstract neural networks, which do not necessarily represent networks of actual neurons (in fact, a processing element in a connectionist network may represent an entire brain area rather than an actual neuron); finally, computational neuroscientists tend to offer explanations in terms of mathematical models that represent concrete neural networks based on neurophysiological evidence. Explanations at different mechanistic levels are not necessarily in conflict with each other, but they do need to be integrated to describe a multi-level mechanism (Craver, 2007). Integrating explanations at different levels into a

unified multi-level mechanistic picture may require revisions in the original explanations themselves.

Different parties in the dispute between classicism, connectionism, and computational neuroscience may offer different accounts of how the different levels relate to one another. As I pointed out, one traditional view is that computational explanations are not even pitched at a mechanistic level. Instead, computational and mechanistic explanations belong at two independent levels (Fodor, 1975; Marr, 1982). This suggests a division of labor: computations are the domain of psychologists, while the implementing neural mechanisms are the business of neuroscientists. According to this picture, the role of connectionists and computational neuroscientists is to discover how neural mechanisms implement the computations postulated by (classicist) psychologists.

This traditional view has been criticized as unfaithful to scientific practices. It's been pointed out that (i) both psychologists and neuroscientists offer computational explanations (e.g., Piccinini, 2006), (ii) far from being independent, different levels of explanation constrain one another (e.g., Feest, 2003), and (iii) both computational explanations (Churchland and Sejnowski, 1992) and mechanistic explanations (Craver, 2007) can be given at different levels.

One alternative to the traditional view is that connectionist or neurocomputational explanations simply replace classicist ones. Perhaps some connectionist computations approximate classical ones (Smolensky, 1988), or perhaps not. In any case, some authors maintain that classicist constructs, such as program execution, play no causal role in cognition and will be eliminated from cognitive science (e.g., Churchland, 2007).

A more neutral account of the relation between explanations at different levels is provided by the mechanistic account of computation. According to the mechanistic account, computational explanation is just one type of mechanistic explanation. Mechanistic explanations provide components with such properties and organization that they produce the explanandum. Computational explanation, then, is explanation in terms of computing mechanisms and components—mechanisms and components that perform computations. (Computation, in turn, is the manipulation of medium-independent vehicles.) Mechanistic explanations come with many levels of mechanisms, where each level is constituted by its components and the way they are organized. If a mechanistic level produces its behavior by the action of computing components, it counts as a computational level. Thus, a mechanism may contain zero, one, or many computational levels depending on what components it has and what they do. Which types of computation are performed at each level is an open empirical question to be answered by studying cognition and the nervous system at all levels of organization.

## **8. Conclusion**

Computationalism is here to stay. We have every reason to suppose that cognitive capacities have computational explanations, at least in the generic sense that they can be explained in terms of the functional processing of medium-independent vehicles. Moreover, everyone is (or should be) a connectionist or computational neuroscientist, at least in the general sense of embracing neural computation. Nonetheless, much work remains to be done. More needs to be said about how different levels of explanation relate to one another; and how, in particular, computational explanations in psychology

and neuroscience relate to each other, and to other kinds of explanation. Ultimately, the computational study of cognition will require that we integrate different mechanistic levels into a unified, multi-level explanation of cognition.

Similarly, much work remains to be done to characterize the specific computations on which cognition depends: whether we ought to be committed to strong or to weak associationism; whether—and to what extent—the satisfactory explanation of cognition requires classical computational mechanisms as opposed to non-classical digital computation; and whether we need to invoke processes that involve non-digital computation, as some have maintained (Figure 3). It may turn out that one computational theory is right about all of cognition or it may be that different cognitive capacities are explained by different kinds of computation. To address these questions in the long run the only effective way is to study nervous systems at all its levels of organization and find out how they exhibit cognitive capacities.

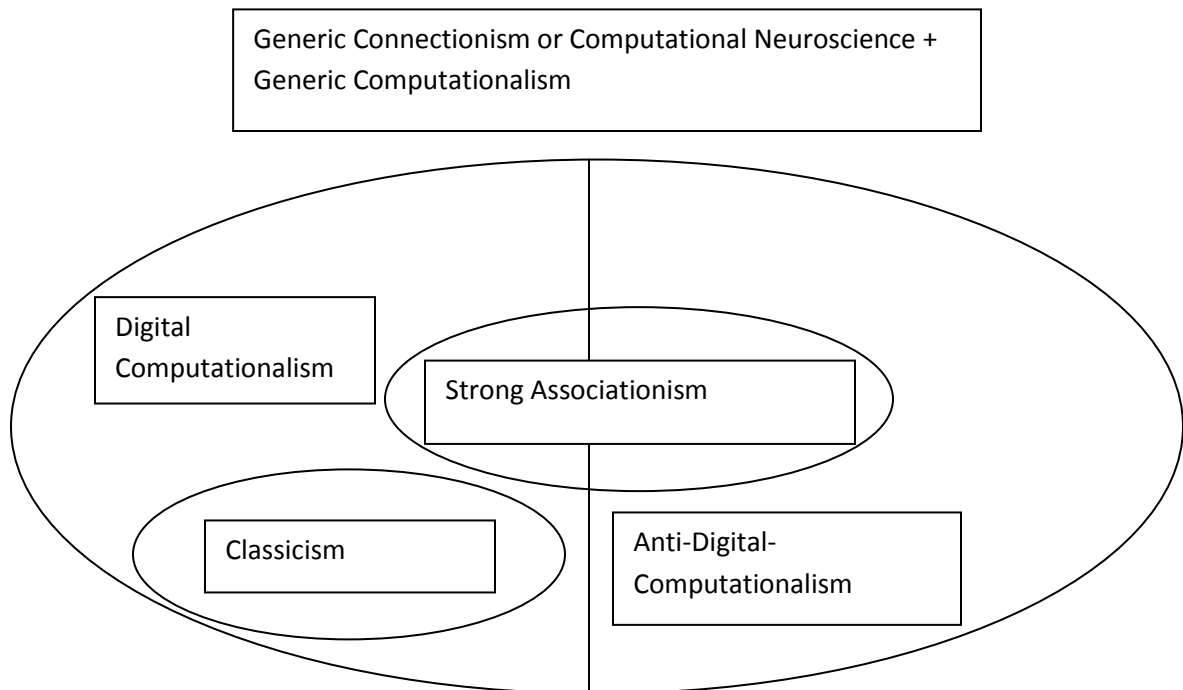


Figure 3. Some prominent forms of computationalism and their relations.

**Acknowledgments.** Parts of this essay are revised and adapted descendants of parts of Piccinini 2009, 2010a, and Piccinini and Scarantino 2010. Thanks to Susan Schneider, Eric Thomson, and the editors for helpful comments and to James Virtel for editorial assistance. This material is based in part upon work supported by the National Science Foundation under Grant No. SES-0924527.

## References

- Adrian, E. D. (1928). *The Basis of Sensation: The Action of the Sense Organs*. New York, Norton.
- Aizawa, K. (2003). *The Systematicity Arguments*. Boston, Kluwer.
- Anderson, J. R. (2007). *How Can the Human Mind Occur in the Physical Universe?* Oxford, Oxford University Press.
- Baars, B. J., W. P. Banks, and J. B. Newman, Eds. (2003). *Essential Sources in the Scientific Study of Consciousness*. Cambridge, MA, MIT Press.
- Bechtel, W. and A. Abrahamsen (2002). *Connectionism and the Mind: Parallel Processing, Dynamics, and Evolution in Networks*. Malden, MA, Blackwell.
- Block, N. (1978). Troubles with Functionalism. *Perception and Cognition: Issues in the Foundations of Psychology*. C. W. Savage. Minneapolis, University of Minnesota Press. 6.
- Boden, M. A. (2006). *Mind as Machine: A History of Cognitive Science*. Oxford, Oxford University Press.

- Bringsjord, S. and K. Arkoudas (2007). On the Provability, Veracity, and AI-Relevance of the Church-Turing Thesis. *Church's Thesis After 70 Years*. A. Olszewski, J. Wolenski and R. Janusz. Heusenstamm, Ontos.
- Caianiello, E. R. (1961). "Outline of a Theory of Thought Processes and Thinking Machines." *Journal of Theoretical Biology* **1**.
- Chalmers, D. J. (1994). "On Implementing a Computation." *Minds and Machines* **4**.
- Chalmers, D. J. (1996). "Does a Rock Implement Every Finite-State Automaton?." *Synthese* **108**.
- Church, A. (1936). "An Unsolvable Problem in Elementary Number Theory." *The American Journal of Mathematics* **58**.
- Churchland, P. M. (2007). *Neurophilosophy at Work*. Cambridge, Cambridge University Press.
- Churchland, P. S. and T. J. Sejnowski (1992). *The Computational Brain*. Cambridge, MA, MIT Press.
- Copeland, B. J. (2000). "Narrow Versus Wide Mechanism: Including a Re-Examination of Turing's Views on the Mind-Machine Issue." *The Journal of Philosophy* **XCVI**(1).
- Copeland, B. J. and Proudfoot, D. (1996). "On Alan Turing's Anticipation of Connectionism". *Synthese* **113**.
- Craver, C. F. (2007). *Explaining the Brain*. Oxford, Oxford University Press.
- Cummins, R. (1983). *The Nature of Psychological Explanation*. Cambridge, MA, MIT Press.
- Davis, M. D., R. Sigal, and E. J. Weyuker (1994). *Computability, Complexity, and Languages*. Boston, Academic.

- Dayan, P. and L. F. Abbott (2001). *Theoretical Neuroscience: Computational and Mathematical Modeling of Neural Systems*. Cambridge, MA, MIT Press.
- Dennett, D. C. (1991). *Consciousness Explained*. Boston, Little, Brown & Co.
- Dretske, F. I. (1981). *Knowledge and the Flow of Information*. Cambridge, MA, MIT Press.
- Edelman, G. M. (1992). *Bright Air, Brilliant Fire: On the Matter of the Mind*. New York: Basic Books.
- Egan, F. (1995). "Computation and Content." *Philosophical Review* **104**.
- Eliasmith, C. and C. H. Anderson (2003). *Neural Engineering: Computation, Representation, and Dynamics in Neurobiological Systems*. Cambridge, MA, MIT Press.
- Ermentrout, G. B. and D. H. Terman (2010). *Mathematical Foundations of Neuroscience*. New York, Springer.
- Feest, U. (2003). "Functional Analysis and the Autonomy of Psychology." *Philosophy of Science* **70**.
- Feldman, J. A. and D. H. Ballard (1982). "Connectionist Models and their Properties." *Cognitive Science* **6**.
- Fodor, J. A. (1968). "The Appeal to Tacit Knowledge in Psychological Explanation." *Journal of Philosophy* **65**.
- Fodor, J. A. (1975). *The Language of Thought*. Cambridge, MA, Harvard University Press.
- Fodor J.A. (1980). "Methodological Solipsism Considered as a Research Strategy in Cognitive Psychology." *Behavioral and Brain Sciences* **3**(1).

- Fodor, J. A. (1981). "The Mind-Body Problem." *Scientific American* **244**.
- Fodor, J. A. (1983). *The Modularity of Mind*. MIT Press: Cambridge, MA.
- Fodor, J. A. (2008). *LOT 2: The Language of Thought Revisited*. Oxford, Oxford University Press.
- Fodor, J. A. and Z. W. Pylyshyn (1988). "Connectionism and Cognitive Architecture." *Cognition* **28**.
- Freeman, W. J. (2001). *How Brains Make Up Their Minds*. New York: Columbia University Press.
- Gallistel, C. R. and A. P. King (2009). *Memory and the Computational Brain: Why Cognitive Science Will Transform Neuroscience*. Malden, Wiley-Blackwell.
- Garson, J. (2003). "The Introduction of Information into Neurobiology". *Philosophy of Science* **70**.
- Gerard, R. W. (1951). Some of the Problems Concerning Digital Notions in the Central Nervous System. *Cybernetics: Circular Causal and Feedback Mechanisms in Biological and Social Systems. Transactions of the Seventh Conference*. H. v. Foerster, M. Mead and H. L. Teuber. New York, Macy Foundation.
- Globus, G. G. (1992). "Towards a Noncomputational Cognitive Neuroscience." *Journal of Cognitive Neuroscience* **4**(4).
- Glymour, C. (1991). "Freud's Androids". J. Neu (ed.), *The Cambridge Companion to Freud*, Cambridge: Cambridge University Press.
- Godfrey-Smith, P. (2009). "Triviality Arguments Against Functionalism." *Philosophical Studies* **145**(2).
- Grice, H. P. (1957). "Meaning." *The Philosophical Review* **66**(3).

- Harman, G. (1973). *Thought*. Princeton, Princeton University Press.
- Hebb, D. O. (1949). *The Organization of Behavior: A Neuropsychological Theory*. New York, Wiley.
- Hodgkin, A. L., & Huxley, A. F. (1952). "A Quantitative Description of Membrane Current and Its Application to Conduction and Excitation in Nerve." *The Journal of Physiology* **117**.
- Hopfield, J. J. (1982). "Neural Networks and Physical Systems with Emergent Collective Computational Abilities." *Proceedings of the National Academy of Sciences* **79**.
- Horgan, T., and Tienson, J. (1996). *Connectionism and the Philosophy of Psychology*. Cambridge, MA: MIT Press.
- Horst, S.W. (1996). *Symbols, Computation, and Intentionality: A Critique of the Computational Theory of Mind*. Berkeley, CA: University of California Press.
- Householder, A. S. and H. D. Landahl (1945). *Mathematical Biophysics of the Central Nervous System*. Bloomington, Principia.
- Kleene, S. C. (1952). *Introduction to Metamathematics*. Princeton, Van Nostrand.
- Kleene, S. C. (1956). Representation of Events in Nerve Nets and Finite Automata. *Automata Studies*. C. E. Shannon and J. McCarthy. Princeton, NJ, Princeton University Press.
- Klein, C. (2008). "Dispositional Implementation Solves the Superfluous Structure Problem." *Synthese* **165**.
- Lapicque, L. (1907/2007). "Quantitative Investigation of Electrical Nerve Excitation Treated as Polarization." *Biological Cybernetics* **97**: 341-349. Translation from the French by N. Brunel and M. van Rossum.

- Leff, H. S. and A.F. Rex, Ed. (2003). *Maxwell's Demon 2: Entropy, Classical and Quantum Information, Computing*. Bristol, Institute of Physics Publishing.
- Knight, B. W. (1972). "Dynamics of Encoding in a Population of Neurons." *Journal of General Physiology* **59**.
- Lycan, W. (1987). *Consciousness*. Cambridge, MA, MIT Press.
- Machamer, P. K., Darden, L., and C. Craver. (2000). "Thinking About Mechanisms." *Philosophy of Science* **67**.
- Marr, D. (1982). *Vision*. New York, Freeman.
- McCulloch, W. S. and W. H. Pitts (1943). "A Logical Calculus of the Ideas Immanent in Nervous Activity." *Bulletin of Mathematical Biophysics* **7**.
- Miller, G. A., E. H. Galanter, et al. (1960). *Plans and the Structure of Behavior*. New York, Holt.
- Newell, A. (1980). "Physical Symbol Systems." *Cognitive Science* **4**.
- Newell, A. (1990). *Unified Theories of Cognition*. Cambridge, MA, Harvard University Press.
- Newell, A. and H. A. Simon (1976). "Computer Science as an Empirical Enquiry: Symbols and Search." *Communications of the ACM* **19**.
- O'Reilly, R. C. and Y. Munakata (2000). *Computational Explorations in Cognitive Neuroscience: Understanding the Mind by Simulating the Brain*. Cambridge, MA, MIT Press.
- Penrose, R. (1994). *Shadows of the Mind*. Oxford, Oxford University Press
- Perkel, D. H. (1990). "Computational Neuroscience: Scope and Structure." In E. L. Schwartz, *Computational Neuroscience*. Cambridge, MA: MIT Press.

- Piccinini, G. (2003). "Alan Turing and the Mathematical Objection." *Minds and Machines* **13**(1).
- Piccinini, G. (2004). "Functionalism, Computationalism, and Mental Contents." *Canadian Journal of Philosophy* **34**(3).
- Piccinini, G. (2006). "Computational Explanation in Neuroscience." *Synthese* **153**(3).
- Piccinini, G. (2007a). "Computational Modeling vs. Computational Explanation: Is Everything a Turing Machine, and Does It Matter to the Philosophy of Mind?." *Australasian Journal of Philosophy* **85**(1).
- Piccinini, G. (2007b). "Computing Mechanisms." *Philosophy of Science* **74**(4).
- Piccinini, G. (2008a). "Computation without Representation." *Philosophical Studies* **137**(2).
- Piccinini, G. (2008b). "Computers." *Pacific Philosophical Quarterly* **89**(1).
- Piccinini, G. (2008c). "Some Neural Networks Compute, Others Don't." *Neural Networks* **21**(2-3).
- Piccinini, G. (2009). "Computationalism in the Philosophy of Mind." *Philosophy Compass* **4**(3).
- Piccinini, G. (2010a). Computation in Physical Systems. *Stanford Encyclopedia of Philosophy (Fall 2010 Edition)*. E. N. Zalta (ed.). URL = <http://plato.stanford.edu/archives/fall2010/entries/computation-physicalsystems/>.
- Piccinini, G. (2010b). "The Mind as Neural Software? Understanding Functionalism, Computationalism, and Computational Functionalism." *Philosophy and Phenomenological Research* **81**(2).

- Piccinini, G. and A. Scarantino (2010). "Information Processing, Computation, and Cognition." *Journal of Biological Physics* (in press).
- Pinker, S. (1997). *How the Mind Works*. New York, Norton.
- Port, R. and T. van Gelder, Eds. (1995). *Mind as Motion: Explorations in the Dynamics of Cognition*. Cambridge, MA, MIT Press.
- 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**.
- Putnam, H. (1988). *Representation and Reality*. Cambridge, MA: MIT Press.
- Pylyshyn, Z. W. (1984). *Computation and Cognition*. Cambridge, MA, MIT Press.
- Rashevsky, N. (1938). *Mathematical Biophysics: Physicomathematical Foundations of Biology*. Chicago, University of Chicago Press.
- Rashevsky, N. (1940). *Advances and Applications of Mathematical Biology*. Chicago, University of Chicago Press.
- Rey, G. (2005). "Mind, Intentionality and Inexistence: An Overview of My Work." *Croatian Journal of Philosophy* **5**(15).
- Rosenblatt, F. (1958). "The Perceptron: A Probabilistic Model for Information Storage and Organization in the Brain." *Psychological Review* **65**.
- Rubel, L. A. (1985). "The Brain as an Analog Computer." *Journal of Theoretical Neurobiology* **4**.
- Rumelhart, D. E., J. M. McClelland, et al. (1986). *Parallel Distributed Processing: Explorations in the Microstructure of Cognition*. Cambridge, MA, MIT Press.

- Rusanen, A.-M. and O. Lappi (2007). The Limits of Mechanistic Explanation in Neurocognitive Sciences. *Proceedings of the European Cognitive Science Conference 2007*. S. Vosniadou, D. Kayser and A. Protopapas (eds.), Lawrence Erlbaum Associates.
- Samuels, R. (2010). "Classical computationalism and the many problems of cognitive relevance". *Studies in History and Philosophy of Science*, **41**(3).
- Scheutz, M. (1999) "When Physical Systems Realize Functions ... ." *Minds and Machines* **9**(2).
- Scheutz, M. (2001). "Causal versus Computational Complexity." *Minds and Machines* **11**(4).
- Schwartz, E. L. (1990). *Computational Neuroscience*. Cambridge, MA, MIT Press.
- Searle, J. R. (1980). "Minds, Brains, and Programs." *The Behavioral and Brain Sciences* **3**.
- Searle, J. R. (1992). *The Rediscovery of the Mind*. Cambridge, MA, MIT Press.
- Shagrir, O. (2001). "Content, Computation and Externalism". *Mind* **110**(438).
- Shagrir, O. (2006). "Why We View the Brain as a Computer." *Synthese* **153**(3).
- Shannon, C. E., and W. Weaver, 1949, *The Mathematical Theory of Communication*, Urbana: University of Illinois Press.
- Siegelmann, H. T. (2003). "Neural and Super-Turing Computing." *Minds and Machines* **13**(1).
- Smolensky, P. (1988). "On the Proper Treatment of Connectionism." *Behavioral and Brain Sciences* **11**.

- Smolensky, P. & Legendre, G. (2006). *The Harmonic Mind: From Neural Computation to Optimality-Theoretic Grammar. Vol. 1: Cognitive Architecture; Vol. 2: Linguistic and Philosophical Implications*. Cambridge, MA: MIT Press.
- Spivey, M. (2007). *The Continuity of Mind*. Oxford, Oxford University Press.
- Sprevak, M. (2010). "Computation, individuation, and the received view on representation." *Studies in History and Philosophy of Science*, **41**(3).
- Stein, R. (1965). "A Theoretical Analysis of Neuronal Variability." *Biophysical Journal* **5**.
- Stich, S. (1983). *From Folk Psychology to Cognitive Science*. Cambridge, MA: MIT Press.
- Thompson, E. (2007). *Mind in Life: Biology, Phenomenology, and the Sciences of Mind*. Cambridge, MA, Harvard University Press.
- Thorndike, E. L. (1932). *The Fundamentals of Learning*. New York, Teachers College, Columbia University.
- Turing, A. M. (1936-7 [1965]). On computable numbers, with an application to the Entscheidungsproblem. *The Undecidable*. M. Davis. Ewlett, Raven.
- Turing, A. M. (1948). "Intelligent Machinery." Reprinted in D. Ince (ed.), *Mechanical Intelligence*. Amsterdam, North-Holland.
- van Gelder, T. (1995). "What Might Cognition Be, if not Computation?" *The Journal of Philosophy* **XCII**(7).
- van Gelder, T. (1998). "The Dynamical Hypothesis in Cognitive Science." *Behavioral and Brain Sciences* **XXI**.

von Neumann, J. (1945). First Draft of a Report on the EDVAC. Philadelphia, PA, Moore School of Electrical Engineering, University of Pennsylvania.

Wilson, H. R. and J. D. Cowan (1972). "Excitatory and Inhibitory Interactions in Localized Populations of Model Neurons." *Biophysical Journal* **12**.

Wilson, R. A. (1994). "Wide Computationalism." *Mind* **103**.

Wimsatt, W. C. (2002). Functional Organization, Analogy, and Inference. *Functions: New Essays in the Philosophy of Psychology and Biology*. A. Ariew, R. Cummins, and M. Perlman. Oxford, Oxford University Press.

---

<sup>i</sup> The *limited* pancomputationalism entailed by the causal account should not be confused with the *unlimited* pancomputationalism according to which everything implements *every* computation (Putnam, 1988; Searle, 1992). Unlimited pancomputationalism is a reductio ad absurdum of a naïve account of computation according to which any mapping from a computational description to a physical description of a system is enough to ascribe a computation to the system. The accounts of computation that are discussed in the text are intended to avoid unlimited pancomputationalism.

<sup>ii</sup> Nevertheless, notice that in the connectionist literature the internal processes of I/O digital systems are often called ‘algorithmic’, using an extended sense of the term ‘algorithm’.

<sup>iii</sup> There are other aspects of spikes that are thought to be functionally relevant at least in some cases, but such other aspects are still medium-independent.

---

<sup>iv</sup> The halting problem is the problem of determining whether any given Turing machine will ever halt while computing on any given input. There is no Turing machine that solves the halting problem.