Review of *Physical Computation: A Mechanistic Account*

Computers have changed the world dramatically over the last half century or so. They have revolutionized the way we work, study, take leisure, communicate, and socialize. But what is the nature of concrete, physical, computation? What does it mean to say that a physical system computes? Quite surprisingly, there is little agreement among philosophers and practitioners regarding this question. Even more strikingly, the well-established mathematical theory of computability, started with Gödel, Church, Turing, and others, does not provide a good answer to this question. The theory focuses on what and how can be computed by means of an algorithm (effectively). It even tells us how to build universal machines – our ordinary digital computers – that can compute any function that is computable by an algorithm. But it is highly questionable that the mathematical theory of computability can account for the wide variety of computing systems we encounter today, such as quantum computing, analog computing, DNA computing, hypercomputing (computing functions that are not Turing computable), and so on. The mathematical theory of computation does not even tell us whether and how nervous systems perform computations, and in what sense. This, however, is certainly no flaw of the theory given that it was never intended to be a theory of physical computation (see Copeland, Dresner, Proudfoot and Shagrir 2017).

In his PhD dissertation, Gualtiero Piccinini (2003) directs the debate over the nature of physical computation into new routes. He proposes to locate physical computation within the mechanistic framework in philosophy of science. This framework emphasizes the centrality of the so-called mechanistic explanations in the sciences, especially in biology and neuroscience. A mechanistic explanation of a system appeals to its components, their functions, and their organization (The *locus classicus* is usually identified with Bechtel and Richardson 1993/2010; Machamer, Darden and Craver 2000; Glennan 2002; and Craver 2007). According to Piccinini, a computational explanation is a special case of mechanistic explanation. A physical computing system is a mechanism “whose teleological function is performing a physical computation. A physical computation is the manipulation (by a functional mechanism) of a medium-independent vehicle according to a rule.” (2015: 10) Piccinini has developed the account in a series of papers in which he substantially expands, and significantly modifies, the original proposal. Over the years his account has attracted attention and convinced others to propose mechanistic accounts of computation (see, e.g., Kaplan 2011; Milkowski 2013; Fresco 2014). The mechanistic account today is perhaps the “received view” about physical computation.

Physical computation: A mechanistic account systematically integrates the various components and adds new, essential, ingredients to complete an overall picture. It would be no exaggeration to say that *Physical computation* is one of the best books written on the conceptual foundations of computation. It is broad in the range of issues covered; it is very detailed in its analysis and argumentation; it makes many vital and novel distinctions that have been overlooked in the debate; it carefully covers both past and recent literature; and, as I emphasize above, it offers an original and bold account of physical computation.

In what follows I provide a brief survey of *Physical computation*. I then make few comments about the integration of physical computation within the mechanistic framework.

1. A short tour through *Physical computation*

Physical computation starts with a list of desiderata of an account of physical computation (Chapter 1). *Objectivity* is the demand that “whether a system performs a particular computation is a matter of fact” (p. 11). *Explanation* is the demand to account for the explanatory role of computation, namely, “how appeals to computation explain the behavior of computing systems” (p. 12). Next is the demand that the account correctly classifies computing and non-computing systems. The demand is divided into two desiderata. *The right things compute* says that an account should classify as computing systems such as calculators, digital and analog computers, and artificial and natural neural networks. *The wrong things don’t compute* says that an account should classify as non-computers paradigmatic examples such as planetary systems, hurricanes, and digestive systems. *Miscomputation* is the demand to account for miscomputing a function. *Taxonomy* requires the accounting for sameness and difference in kinds of computation.

Chapters 2 to 4 concern extant accounts. Chapter 2 is about mapping accounts, which assert that a physical system computes if there is a mapping from a set of states and relations of the physical system onto the states and relations of an abstract computation (e.g., an automaton). A challenge to simple mapping accounts is the Putnam/Searle triviality results which roughly say that we can find such mapping between every physical system and every abstract computation. This would imply that rocks, chairs and every other physical system compute (Piccinini calls this thesis *limited pancomputationalism*). Moreover, each physical system performs every computation (*unlimited pancomputationalism*). Piccinini argues that unlimited pancomputationalism violates the desideratum of objectivity. Limited pancomputationalism erases the distinction between systems that compute and systems that don't; it thus violates the *wrong things don't compute* desideratum (assuming that rocks and chairs don't compute). More restrictive mapping accounts (causal, counterfactual, dispositional) might avoid unlimited pancomputationalism, but are still committed to limited pancomputationalism. The claims about pancomputationalism are further developed in chapter 4.

Chapter 3 concerns semantic accounts, which assert that computations and their states are individuated in terms of their semantic content. Semantic accounts seem to handle triviality results much better, as they exclude systems that do not represent, e.g., rocks and hurricanes. They also seem to apply well to minds, brains and digital computers that all seem to represent. Nevertheless, Piccinini deems the semantic accounts wrong. For one thing, some physical systems compute though they do not represent. For another, artificial digital computers seem to involve observer-dependent, derivative, content ("interpretational semantics") that is observer-relative, and this feature violates *objectivity*. Piccinini thoroughly reviews and rebuts arguments for the semantic accounts. The arguments often rest on the premise that the explanandum computed functions are individuated semantically. Piccinini points out, however, that this premise cannot establish the semantic accounts, as these functions can also be individuated non-semantically. Piccinini agrees that some arguments for the semantic accounts lead to computational externalism (Shagrir 2001; Rescorla 2013). But he says that the external features pertinent to computational individuation needn't be semantic.

In chapter 5, Piccinini turns to develop his own, mechanistic, account. On the face of it, computation does not fall squarely within the mechanistic framework. Some philosophers (REFS) even contrast computational explanations with mechanistic explanations. They view computational explanations as species of functional analyses and argue that the latter are autonomous and distinct from mechanistic explanations. Functional analyses specify functional properties whereas mechanistic explanations specify structural properties that realize the functions. In this chapter (which is based on the widely discussed Piccinini and Craver (2011) paper) Piccinini argues that functional analyses and mechanistic explanations are tightly related. Functional analyses are *sketches of mechanisms*, in the sense that they omit structural aspects of the mechanism. Filling in these aspects turns a functional analysis into a full-blown mechanistic explanation (p. 75). What about computational explanations? They specify medium-independent properties. Nevertheless, Piccinini argues, they are mechanistic explanations to the extent that they specify relevant structural features such as the structural components that do the processing and their organization (p. 98).

In Chapter 6, Piccinini defines the notion of teleological function that is pertinent to his characterization of computation. The notion applies both to natural and artificial (computing systems). Chapter 7 wraps up the essential ingredients into the definition of concrete physical computation (p. 121), and argues that the account satisfies the list of desiderata mentioned above. It should be noted that the mechanistic account has undergone some serious modifications throughout the years. Early on, the account identified computation with some form of digital mechanism. As such it excluded analog computers (Piccinini 2007: 519-520), some connectionist systems (2007: 518), and some neural networks (2008). At some point, however, Piccinini (e.g. Piccinini and Scarantino 2011) extends his account to other forms of computation. He introduces the very general notion of *generic* *computation*, which includes both analog and digital computation. On this new extended definition all neural networks compute, though some of them perform non-digital computations (2015: 221-223).The main modification is that the vehicles of computation need not refer only to digits, but also to variables or specific values of a variables (2015: 121). Chapters 8-13 apply the mechanistic notion of generic computation to different kinds of computing mechanisms. Chapter 8 is about the primitive components of computing mechanisms, and chapter 9 about complex components. Chapter 10 concerns digital calculators, and chapter 11 digital computers. Chapter 12 is about analog computers, and chapter 13 is about parallel computers and neural networks.

The last three chapters concern more specialized yet topical issues. Chapter 14 (which is based on Piccinini and Scarantino 2011) examines the relations between information and computation. Piccinini argues that systems can compute without processing information. He also demonstrates that different computing systems that undertake information-processing can involve different kinds of information, e.g., non-semantic (Shannon) information, and natural and non-natural semantic information. The highly interesting final two chapters are about the physical Church-Turing thesis, which limits the computational power of physical systems to Turing computability. Piccinini draws an important distinction between bold (Chapter 15) and modest (Chapter 16) theses. The bold physical Church-Turing thesis asserts that the behavior (function) of any physical process is Turing-computable. The modest physical Church-Turing thesis states that any function that is *computed* by a physical process is Turing-computable. Physical systems that violate the modest thesis are known as hypercomputers. Piccinini suspects, however, that there aren’t genuine hypercomputers; the term *genuine* refer to physical systems that satisfy certain *usability constraints*. The book closes with a short epilogue that summarizes the main results of the account.

2. How well does computation integrate within the mechanistic framework?

Physical computation advances a formidable and attractive account of computation. It covers a wide terrain that prompts many points of discussion and controversy. I should say that as a proponent of the semantic stance, I have different views about physical computation. But I will refrain from responding here to the arguments launched by Piccinini against semantic accounts. Instead, I will focus on three issues that pertain to the relationship between physical computation and the mechanistic framework. My aim is not to argue for or against the proposed account, but to point to some issues and questions that deserve further discussion.

The first issue concerns the sense in which Piccinini's account is mechanistic. I raise this because it seems that *mechanistic* plays a different role in Piccinini's account than the role played by *semantic*, *mapping*, and other notions in rival accounts. Semantic and mapping properties play a classificatory role in accounts of computation. They are used to *exclude* certain non-computing processes; they play a role in meeting *the wrong system don't compute* desideratum. According to semantic accounts, processes that do not involve semantic properties do not compute. According to mapping accounts, processes that do not implement (i.e., bear mapping relations to) abstract computations do not compute. The term *mechanism*, however, is not used to exclude systems that do not compute; it plays no role in meeting *the wrong systems don't compute* desideratum. This is simply because the non-computing systems are also mechanisms. The mechanistic account of computation thus aims to distinguish between *computing and non-computing mechanisms*:

> The main challenge for the mechanistic account is to specify properties that distinguish computing mechanisms from other (non-computing) mechanisms – and corresponding to those, features that distinguish computational explanations from other (non-computational) mechanistic explanations. (2015: 120)

What are the properties that distinguish computing mechanisms from other (non-computing) mechanisms? Going back to the definition, we can notice three: The *teleological function* that excludes planetary systems, the weather and many other systems (p. 145); *medium-independence* that exclude cooking, cleaning (p. 122) as well as digestive processes (p. 146-7); and the governing *rule* that excludes random-number generators (p. 147). But we can notice that none of these properties – teleological function, medium-independence, and rule – bears a special relation to the mechanistic framework. They can be, and indeed have been, adopted in non-mechanistic accounts of computation. According to Fodor (1994), computational (i.e., syntactic) properties are conceived as high-order physical properties, and in this sense are medium-independent. Hardcastle (1995) discusses computation in some teleological terms. Copeland (1996) and many others associate computation with a rule (e.g., algorithm).

Another central feature of the mechanistic account is that it is non-semantic (see also Milkowski 2013, and Fresco 2014). But, again, being non-semantic is not a distinctive feature of mechanistic accounts. There are many accounts of computation that are neither mechanistic nor semantic; examples are mapping, and syntactic accounts. Moreover, it seems that an account of computation can in principle be both mechanistic and semantic. For example, we can replace the teleological function, in Piccinini's definition of computation, with a *semantic function*, or at least we need a reason to see why not. All this does not undermine the adequacy of Piccinini's definition. The upshot, rather, is that central features in the account – the teleological function, medium-independence, rule, and being non-semantic – are not by themselves tied to the mechanistic framework. There are mechanistic analyses that lack these features, and non-mechanistic analyses that have these features.

So in what sense is the proposed account mechanistic? Piccinini writes:

> The present account is *mechanistic* because it deems computing systems a kind of functional mechanism – mechanism with teleological functions. Computational explanation – the explanation of a mechanism's capacities in terms of the computations it performs is a species of mechanistic explanation (p. 118).

As I understand it, Piccinini suggests that since computation is a kind of mechanism, it would be natural to analyze computation and computational explanations from within the mechanistic explanatory framework. The mechanistic framework provides the tools to explicate computational explanation, and the features (medium-independence, teleological function, etc.) that define the computing mechanism in general. It thus provides the tools to distinguish between computational explanations and other (non-computational) mechanistic explanations, and, correspondingly, between computing and non-computing mechanisms. If this is correct, then the account is mechanistic because computing systems (much like non-computing systems) are mechanisms, and the mechanistic framework naturally provides the means to account for mechanisms and their (mechanistic) explanations.

The next two issues concern the relations between computational and mechanistic explanations. One concerns the status of computational explanations within the mechanistic framework. Piccinini and Craver 2011 might have given the impression that computational explanations are sketches of mechanisms. This is a reasonable interpretation if we take computational explanations to be species of functional analyses (which are described as sketches). Also, Piccinini and Craver classify Marr's computational and algorithmic levels – the only example of computational explanations in the paper – as sketches. Depicting computational explanations as sketches has one salient advantage. According to this picture, computational explanations nicely integrate within the mechanistic framework, in that computational properties and implementational properties belong to the same level of mechanism. When adding to the computational sketches the missing structural, e.g., implementational, properties we get a full-blown mechanistic explanation of a given phenomenon. But this picture, of computational explanations as sketches, also has a major disadvantage. Computational explanations, as sketches, are weak, partial or elliptical explanations, and this does not seem to do justice to many of them (Haimovici 2013).

The other option is that computational explanations can be full-blown mechanistic explanation; they are full-blown to the extent they refer to relevant functional and structural properties. In *Physical Computation*, Piccinini clearly favors this option: "Computational explanations count as full-blown mechanistic explanations, where structural and functional properties are inextricably mixed" (p. 124). But this option immediately raises a puzzle: Computational explanations refer to the computational, medium-independent, properties of the mechanism. However, full-blown mechanistic explanations must refer to some structural, e.g., implementational, properties. How can an explanation be both computational and full-blown mechanistic? Piccinini's answer is that some computational, medium-independent, properties have structural features; for example, those determining “the type of vehicle being processed (digital, analog, and what have you)” (p. 98). Piccinini does not deny that these properties also have some functional aspects. He rather dismisses the distinction between functional and structural: “There is no such thing as a purely functional component or a purely functional property” (p. 99).

This answer certainly seems to be in the right direction. One might wonder, however, whether it does not bring back the claim that computational explanations are distinct. Yes, computational explanations are full-blown mechanistic ones, but they are nevertheless distinct from implementational mechanistic explanations. The first refers to medium-independent (functional and structural) properties, whereas the latter refers to medium-dependent, implementational, properties. In other words, we can reformulate the distinctness thesis around the medium-independent/medium-dependent distinction instead of the dismissed functional/structural distinction. It is the first distinction that supports the thesis that computational explanations are distinct (and perhaps autonomous) from the implementational levels.

One might also wonder how the computational and implementational levels fit together, within the mechanistic hierarchy. Piccinini himself highlights the problematics when referring to Marr’s renowned tri-level framework: “His ‘levels’ are not levels of mechanisms because they do not describe component/sub-component relations. The algorithm is not a component of the computation, and the implementation is not a component of the algorithm” (p. 98). Indeed, the realization relation – of medium-independent properties by some implementational properties – is *not* a part/whole relationship. The ‘0’s and ‘1’s might be implemented by certain specific voltages, but the voltages are not parts of the ‘0’s and ‘1’s. The problem can be resolved when computational explanations are conceived as sketches (the first option), but it is less clear how full-blown computational explanations relate to the implementational levels within the picture of multi-level mechanism.

The last issue concerns more direct challenges to the thesis that computational explanations are mechanistic explanations. Piccinini says that “the main challenge for the mechanistic account is to specify … features that distinguish computational explanations from other (non-computational) mechanistic explanations”. However, there are many who dispute the assumption that all computational explanations fall within the mechanistic framework (even if computations are mechanisms). There are three kinds of objections to this assumption, all aiming to show that at least some computational explanations do not conform to the norms of mechanistic explanations. The most widespread objection is that abstract, including computational, explanations can be full-blown (i.e., not sketches), even if making no reference to structural properties at all (Weiskopf 2011; Levy and Bechtel 2013; Barrett 2014; Shapiro 2016; see in particular Chirimuuta 2014 and Egan 2017 who specifically discuss computational explanations). As just mentioned, Piccinini agrees that abstract explanations can be full-blown to the extent that they specify the relevant properties (see also Boone and Piccinini 2016); with respect to computational explanations the relevant medium-independent properties place structural constraints on any mechanism that implements them. Some insist, however, that placing structural constraints does not make the properties structural and the explanation mechanistic. Every abstract explanation is constrained to some extent by implementational details (Shapiro 2016). Moreover, while implementational properties can serve as evidence to distinguish between and support candidate explanatory models, this does not make them an integral part of the explanation (Weiskopf 2011; Shapiro 2016).

A second criticism has been that some abstract explanations do not involve componential analysis. They do not decompose the explanandum capacities into sub-components and their organization. Rathkopf (2015) argues that some network models provide non-decompositional explanations for non-decomposable systems, where part-whole decomposition is not possible (see also Weiskopf 2011 who argues for a similar point about noncomponential models in cognitive science; Huneman 2010 who argues that in some cases the explanation does not appeal to causal structure, but to the topological or network properties of the system; and Levy 2013 who argues that decomposition (and localization) plays a lesser role in population genetics, ecology and other macro-biological populational disciplines). The claim does not directly target computational explanations, but Rathkopf in his discussion also refers to computing systems. Piccinini (p. 84) reminds us that the components of computation need not be spatially localized. This seems to be certainly true. The worry, however, is that if we start relaxing the notion of componential analysis, it becomes less attractive to couple computational and mechanistic explanations.

The last objection is that at least part of computational theory in cognitive neuroscience aim to answer certain *why* questions whose explanations do not track causal relations in the mechanistic sense: "It departs fully from the [Kaplan's] model-to-mechanism mapping framework that has been proposed as the criterion for explanatory success" (Chirimuuta 2014). Chirimuuta (2014) locates these *why* questions in the so-called interpretative models which "use computational and information-theoretic principles to explore the behavioral and cognitive significance of various aspects of nervous system function, addressing the question of why nervous systems operate as they do" (Dayan and Abbott 2001). She argues that answering these questions involves explanations which typically make reference to efficient coding principles. Bechtel and Shagrir (2015; Shagrir and Bechtel 2017) choose to associate these *why* questions with the *why* component of computational-level theories whose goal is to demonstrate the basis of the computed function in the physical world (Marr 1982). They claim that the answer consists in some modelling relations between the nervous system and the environment of the system. Rusanen and Lapi (2016) also associate the *why* questions with Marr's computational-level theories, and argue that computational theories provide explanations that express formal, non-causal, dependencies. The mechanists might reply that these *why* questions belong to the "context level" that is not computational (Milkowski 2013), or perhaps try to account for them in terms of the (teleological) function that has utility for the organism.

In sum, *Physical Computation* provides in my view the most comprehensive, detailed and forceful account of physical computation to date. But as I see it, the account is in no way the last word on physical computation. Rather, it paves the way to further, more focused and more informative, discussion. Some (myself included) will ultimately defend mapping, semantic and other accounts, responding to Piccinini's arguments. Others will discuss the claims related to pancomputationalism, hypercomputation, the relation between information and computation and more. And yet others will keep arguing about the fit of computation within the mechanistic framework.

References:

Barrett, David. 2014. Functional analysis and mechanistic explanation. *Synthese* 191: 2695–2714.

Bechtel, William, and Richardson, Robert C. (1993/2010). *Discovering complexity: Decomposition and localization as strategies in scientific research* (2nd edition). MIT Press.

Bechtel, William, and Shagrir, Oron. 2015. The non-redundant contributions of Marr's three levels of analysis for explaining information processing mechanisms. *Topics in Cognitive Science* (*TopiCS*) 7: 312–322.

Boone, Worth, and Piccinini, Gualtiero. 2016. Mechanistic abstraction. *Philosophy of Science* (forthcoming).

Chirimuuta, Mazviita. 2014. Minimal models and canonical neural computations: The distinctness of computational explanation in neuroscience. *Synthese* 191: 127–153.

Copeland, B. Jack. 1996. What is computation? *Synthese* 108: 335–359.

Copeland, B. Jack, Dresner, Eli, Proudfoot, Diane, and Shagrir, Oron. 2017. Time to re-inspect the foundations? *Communications of the ACM* (forthcoming).

Craver, Carl F. 2007. *Explaining the Brain*. Oxford University Press.

Dayan, Peter, and Abbott, Laurence F. 2001. *Theoretical Neuroscience: Computational and Mathematical Modeling of Neural Systems*. MIT Press.

Egan, Frances. 2017. Function-theoretic explanation and the search for neural mechanisms. In *Integrating Psychology and Neuroscience: Prospects and Problems*, edited by David Kaplan. Oxford: Oxford University Press (forthcoming).

Fodor, Jerry A. 1994. *The Elm and the Expert*. MIT Press.

Fresco, Nir. 2014. *Physical Computation and Cognitive Science*. Springer-Verlag.

Glennan, Stuart. 2002. Rethinking mechanistic explanation. *Philosophy of Science* 69: S342–353.

Haimovici, Sabrina. 2013. A problem for the mechanistic account of computation. *Journal of Cognitive Science* 14: 151–81.

Hardcastle, Valerie Gray. 1995. Computationalism. *Synthese* 105: 303–317

Huneman, Philippe. 2010. Topological explanations and robustness in biological sciences. *Synthese* 177: 213–245.

Kaplan, David Michael. 2011. Explanation and description in computational neuroscience. *Synthese* 183: 339–373.

Levy, Arnon. 2013. Three kinds of "new mechanism". *Biology & Philosophy* 28: 99–114.

Levy, Arnon, and Bechtel, William. 2013. Abstraction and the organization of mechanisms. *Philosophy of Science* 80: 241–61.

Machamer, Peter, Darden, Lindley, and Craver, Carl F. 2000. Thinking about mechanisms. *Philosophy of Science* 67: 1–25.

Milkowski, Marcin. 2013. *Explaining the Computational Mind*. MIT Press.

Piccinini, Gualtiero. 2003. *Computations and Computers in the Sciences of Mind and Brain.* Doctoral Dissertation, University of Pittsburgh.

-- 2007. Computing mechanisms. *Philosophy of Science* 74: 501–526.

-- 2008. Some neural networks compute, others don't**.** *Neural Networks* 21: 311–321.

-- 2015. *Physical computation: A mechanistic account*. Oxford University Press.

Piccinini, Gualtiero and Craver, Carl, F. 2011. Integrating psychology and neuroscience: Functional analyses as mechanism sketches. *Synthese* 183: 283–311.

Piccinini, Gualtiero and Scarantino, Andrea. 2011. Information processing, computation, and cognition. *Journal of biological physics* 37:1–38.

Rathkopf, Charles. 2015. Network representation and complex systems. *Synthese (forthcoming). Online version:* doi:10.1007/s11229-015-0726-0.

Rescorla, Michael. 2013. Against structuralist theories of computational implementation. *British Journal for the Philosophy of Science* 64: 681–707.

Rusanen, Anna-Mari, and Lappi, Otto. 2016. On computational explanations. *Synthese (forthcoming). Online version:* doi:10.1007/s11229-016-1101-5.

Shagrir, Oron. 2001. Content, computation, and externalism. *Mind*, 110: 369–400.

Shagrir, Oron, and Bechtel, William. 2017. Marr's computational-level theories and delineating phenomena. In *Integrating Psychology and Neuroscience: Prospects and Problems*, edited by David Kaplan. Oxford University Press (forthcoming).

Shapiro, Lawrence A. 2016. Mechanism or bust ? Explanation in psychology. *British Journal for the Philosophy of Science (forthcoming). Online version:* doi:10.1093/bjps/axv062.

Weiskopf, Daniel A. 2011. Models and mechanisms in psychological explanation. *Synthese* 183: 313–338.