A Problem for the Mechanistic Account of Computation

Sabrina Haimovici

Universidad de Buenos Aires;
National Scientific and Technical Research Council (CONICET)
shaimovici@filo.uba.ar

The mechanistic account of computation proposes that computational explanation is mechanistic, i.e. it explains the behavior and capacities of mechanisms in terms of their components and the functions and organization of those components. According to this account, computing systems are mechanisms that perform computations, that is, they process vehicles according to rules that are sensitive to certain vehicle properties. Despite the emphasis mechanists place on the structural description of mechanisms' components and activities, the description of computations does not rely on the structural individuation of components and activities, but only on their functional individuation. Concrete computations and their vehicles are not described in terms of detailed structural properties, but are instead characterized in a medium-independent way, that is, independently of the physical medium that implements the computation. Thus, the same specific computation can be implemented in different physical media, allowing the multiple realizability of computational systems. In this paper, I argue that the mechanistic account faces a dilemma. If computations and computational systems are individuated in functional terms, then computational explanations are elliptic mechanistic explanations, or mechanism sketches. But, according to mechanists, mechanism sketches are incomplete and explanatorily weak. Alternatively, for the computational explanation to satisfy the criteria for a good mechanistic explanation, we need a new way to individuate computations based on structural properties. However, as a result of this, multiple real-

*I am grateful to Liza Skidelsky, Sergio Barberis, Mariela Destéfano, and Abel Wajnerman Paz for their insightful comments and suggestions on a previous version of this paper. I would also like to thank the participants of the III Workshop of Philosophy and Cognitive Science (Buenos Aires, 2012), especially Gualtiero Piccinini and Miguel Ángel Sebastián, for their helpful discussion and comments. Thanks also to the anonymous referees for their valuable comments.

izability will no longer be possible for computational systems.

Key words: *computation, mechanistic explanation, functional analysis, multiple realizability*

1. Introduction

Computational explanations of cognitive capacities are ubiquitous in psychology and the philosophy of mind. These explanations describe the mind as a computational system: in a first approximation, as a system which processes inputs following certain rules and gives certain outputs. These explanations do not intend to establish mere analogies between minds and computers, but instead to unfold the actual ways in which our minds perform psychological processes. Despite how commonplace computational explanations are in the literature, there is little consensus as to what computations are and what it means for a physical system to perform them, and consequently, as to how we ought to test computational theories and evaluate computational explanations. Piccinini (2012a) maintains that in order to test computational theories of concrete systems, it is necessary to have a clear account of what it means for a physical system to compute. He develops a mechanistic account of computation, according to which the determination of whether or not a system performs computations depends on that system's mechanistic properties (Piccinini 2006, 2007a, 2007b, 2008a, 2008b, 2010, 2012a). Piccinini does not intend to limit his account to computational explanations of psychological capacities, but instead aims to offer an account of computational explanation in general, grounded in the notion of computation established in computability theory and computer sciences (Piccinini 2007a, 2008a). The account of computation in terms of the mechanistic properties of systems can be related to the current attempt to account for explanations from different special sciences under a mechanistic approach (Machamer, Darden & Craver 2000, Craver 2007, Bechtel 2008).

In this paper, I will argue that the mechanistic account of computation faces a dilemma. By adopting the mechanistic view of explanation, Piccinini commits himself to certain adequacy requirements on explanations.

A good mechanistic explanation must describe a system's actual causal structure, which requires the identification of its actual functional and structural properties (Machamer, Darden & Craver 2000, Craver 2006, Craver 2007, Piccinini & Craver 2011). The description of computation offered by Piccinini, however, barely appeals to the structural properties of a system. According to Piccinini, computational vehicles are medium-independent, i.e. they do not depend on the particular properties of the physical medium that implements the computation. Thus, the same specific computation can be implemented in different physical media, allowing for multiple realizability of computational systems.[1] I will defend the idea that if computations are individuated in functional terms, then computational explanation is a kind of functional explanation and hence does not satisfy the criteria for a good mechanistic explanation.

Alternatively, for the computational explanation to be a good mechanistic explanation, we need a new way to individuate computations based on both structural and functional properties. However, multiple realizability would therefore no longer be possible for computational systems. This result would be problematic, since the mechanistic account would fail to reflect some uses of the notion of computation in computer sciences. Moreover, in some cases, the mechanistic account of computation seems to offer evidence of multiple realizability of computations in support of the fact that a given system computes. The mechanistic account of computation must therefore necessarily remain compatible with multiple realizability. The dilemma I am presenting is intended to point out the limits of the mechanistic approach for providing an adequate account of computations and computational explanations. Furthermore, if this approach is not well suited to providing an account of computational explanations, then it does not apply to explanations in cognitive sciences, given that these are paradigmatically computational.

In what follows, I will develop the dilemma presented above. In section 2, I will address the first alternative. I will present the mechanistic account of computation and show that it characterizes computations, computational vehicles, and computing systems mainly in functional terms (section 2.1).

[1] I will use the terms 'realize' and 'implement' interchangeably.

I will argue that, as a consequence, computational explanations can be reconstructed as a kind of functional explanation and that this is problematic since, following the mechanistic normative criteria of adequacy for explanations, functional analyses are mere sketches of mechanisms, not good, full-blown mechanistic explanations (section 2.2). In section 3, I will address the second alternative. I will analyze the consequences entailed by demanding a full, detailed structural description of the components of computing systems. I will argue that in such a case, computations will no longer be multiply realizable (section 3.1) and, as a consequence thereof, an important explanatory attribute of computational explanations will be lost (section 3.2).

2. The Mechanistic Account of Computation: A Functional Characterization of Computing Systems

In an abstract sense, a computation is a formal specification of how to solve or execute certain functions. The primary concern of the mechanistic account of computation is to explain what it means for a concrete physical system to perform or implement computations. Computations can be abstractly defined by mathematical formalisms, such as algorithms or Turing machines (Piccinini 2012a), which describe how to execute functions (input-output mappings). The goal of the mechanistic account is to build a bridge between formal abstract descriptions of computation and the functions and activities of concrete physical systems in such a way that it provides the means to empirically distinguish whether a given system is computational or not. In this section, I will present the mechanistic account of computation and show that computational explanations are given mainly in functional terms (section 2.1). I will argue that under mechanistic criteria of adequacy of explanations, this renders computational descriptions of mechanisms explanatorily weak (section 2.2).

2.1 The Mechanistic Account of Concrete Computations

Piccinini (2006, 2007a, 2007b, 2008a, 2008b, 2010, 2012a) proposes characterizing concrete computations (computations performed by concrete systems) in terms of a system's mechanistic properties. This position embraces the mechanistic view of explanation, according to which a scientific explanation of a phenomenon is a description of its underlying mechanism in terms of its components, its activities, and their organization (Machamer, Darden & Craver 2000, Craver 2007). Mechanisms have parts or components which perform certain activities and are organized in a particular way, and the behavior of mechanisms can be explained in terms of those three aspects. One of the central tenets of the mechanistic view holds that components should be actual spatiotemporal parts of mechanisms and should be individuated by both their structural and functional properties. As Piccinini and Craver (2011) point out, structural properties include, *inter alia*, the size, shape, location, and orientation of objects. Functional properties include the capacity or disposition to perform certain activities and "are specified in terms of effects on some medium or component under certain conditions" (p. 291). For example, the primary sequence of a gene is a structural property, while encoding an episodic memory is a functional one.

The mechanistic approach holds that computing systems are mechanisms. As Piccinini (2007a) explains, computing systems have components (input devices, processing components, and output devices, *inter alia*), each of which has specific functions (respectively, turning external stimuli into computational vehicles, taking computational vehicles as inputs and returning others as outputs following fixed rules, and yielding an output to the environment). The organized execution of such functions by the components can account for the behavior of the system. In this sense, computing systems exhibit the properties of mechanisms.

What makes a mechanism computational? There are different kinds of computations, such as digital and analog, but all of them share a central feature which allows for characterization of computations in a *generic* way. Generic computation is the most comprehensive kind in that it includes different kinds of computations, such as digital, analog, and quantum computation.[2] For a given system to be a computing system in the generic sense, it

[2] Digital computations are defined by manipulations of discrete vehicles—i.e. unambiguously distinguishable states—, while analog computations are defined by manipulations of continuous variables. There are other kinds of computation, such as quantum computation, which is defined as the manipulation of quantum states:

must perform computations; that is, it must process vehicles "according to rules that are sensitive to certain vehicle properties and, specifically, to differences between different portions of the vehicles" (Piccinini & Scarantino 2011, p. 10). This characterization appeals to both rules and computational vehicles.

The rules are mappings from inputs to outputs. The appeal to rules assimilates the classic way of understanding computations as executions of algorithms: taking certain inputs, executing sequences of instructions, and giving certain outputs (Turing 1936, Copeland 1996). However, Piccinini's proposal is not meant to be limited to the classic notion of computation, because the classic notion applies only to algorithmic computation—a kind of digital computation (Piccinini & Scarantino 2011, Piccinini 2012a). Piccinini proposes adopting a more comprehensive notion that applies to all kinds of computation. In this sense, the rules that define the computations need not be specifically algorithms, but only definitions of functions; they could, for example, be systems of differential equations. Since the mechanistic account is concerned with the issue of what makes a concrete physical system computational, it combines the rather abstract notion of input-output mapping with specifications of how those mappings are physically executed or implemented. This is how the vehicles of computation enter into the picture.

In physical computational systems, the rules that define the computations are sensitive to differences in certain physical variables that serve as computational vehicles. In order to compute, a system must be able to undergo certain changes; it must be able to be in different states. The physical variables that are functionally relevant to individuate those states implement computational vehicles. Those variables may vary depending on the system, meaning that different physical systems can implement computational vehicles and rules in various ways inasmuch as the system is able to distin-

qubits. Qubits are different from digital and analog vehicles in that a qubit can take one of the possible basic states plus any superposition thereof (Piccinini 2012a). Piccinini and Bahar (2012) argue in favor of another species of computation, a *sui generis* one, which they claim is the kind of computation that the nervous system implements.

guish between different portions of the computational vehicles or states. For example, in electronic systems (such as our ordinary computers), the implementation of computational rules is based on differences in voltages, and in collision-based systems (such as domino arrangements), it is based on the presence or absence of traveling wave fronts.

However, not every physical system will be able to implement every computation. In order to implement a concrete computation, a physical system needs to exhibit certain degrees of freedom. In other words, it needs to possess a certain number of physical variables that can vary independently from one another. These are the system's dimensions of variation. A system's dimensions of variation, or degrees of freedom, are the number of physical variables that we need to consider in order to define a state of that system. Given the fact that the differences in computational vehicles that are functionally relevant to defining computations can be abstractly characterized, without detailing all the specific physical properties of the vehicles, Piccinini holds that computational vehicles are medium-independent. Computation depends only on certain properties of vehicles, not on every one of their properties. As Piccinini and Scarantino affirm:

> […] 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 (i.e., spatiotemporal parts) of the vehicles along specific dimensions of variation—they are insensitive to any other physical properties of the vehicles. Put yet another way, the rules are functions of state variables associated with certain degrees of freedom that can be implemented differently in different physical media. (2011, p. 8)

Thus, to implement computational vehicles, a system must possess certain physical variables that can exhibit independent variations and thus serve as computational vehicles. There are also further requirements to implement computations. Not only is it necessary that the systems possess the functionally relevant dimensions of variation, but also that those dimensions of variation "can be appropriately accessed and manipulated and that the components of the mechanism are functionally organized in the appropriate way" (Piccinini 2012b, p. 231).

As the previous characterization shows, the mechanistic account of concrete computation imposes some constraints on systems for them to be able to implement computations. These are mainly functional constraints that are based on the dispositions of the components and on what they do. Given a particular system, the degrees of freedom it possesses is one of its structural properties. However, the same degrees of freedom can be implemented in systems that are very different from one another in terms of their other structural properties. Thus, different physical systems that possess the same number of dimensions of variation, that grant their accessibility and manipulation, and also have the relevant functional organization to perform computations, can satisfy those constraints. Therefore, the definition of concrete computations and their vehicles is independent of any particular structural description of a mechanism and allows for the same computation to be implemented in different mechanisms. The mechanistic notion of generic computation thereby leaves open the possibility of multiple realization of computations. The notion of multiple realizability is a highly controversial one. I take it as the idea that a functionally identified kind can be realized in different mechanisms (Weiskopf 2011a). I will return to this notion in section 3.

So far, we have seen that according to Piccinini, computations should be understood mechanistically. For a system to perform computations, it must have certain parts or components which in turn have particular functions, are organized in a certain way, and carry out certain activities that allow the system to compute. In particular, those activities must include processing medium-independent vehicles following rules that are sensitive to certain vehicle properties. This notion of concrete computation is meant to contribute to the evaluation of computational explanations, given that it is common to explain the behavior of certain systems by attributing specific computations to them. Particularly, in the cognitive sciences, it is common to explain psychological capacities as computational processes. Adopting the mechanistic approach, Piccinini proposes characterizing computational explanations in terms of mechanistic explanations, involving the partition of a mechanism into spatiotemporal parts or components, "an assignment of functions and organization to those parts, and a statement that a mechanism's capacities are due to the way the parts and their functions are organized" (Piccinini 2007a, p. 502).

The definition of computations should therefore be based on a mechanistic characterization of the system that performs them. However, as we have seen, computational vehicles are medium-independent and are hence characterized mainly in terms of their functional properties, rather than their structural ones. Computational vehicles are abstract and multiply realizable in different physical media, e.g. mechanical, electromechanical, electronic, or magnetic. Computations, for their part, are characterized as manipulations of medium-independent vehicles according to rules, which is also a functional characterization. Thus, computational explanations omit many structural details. Despite the relevance that mechanism defenders assign to structural properties (cf. Piccinini & Craver 2011), it seems that whether a system realizes a computational system or not depends mainly on its functional properties, rather than on its structural ones. As Piccinini and Craver (2011) point out, there are mutual constraints between functional and structural properties. However, there are no *specific* structural properties besides the possession of the functionally relevant degrees of freedom that are necessary to implement computations, and different physical systems with different structurally individuated components and activities can implement the same computations. Even Piccinini often seems to dispense with structural properties:

> Whether something is a computing system properly so called turns out to depend on whether it has certain functional properties, so that determining whether the mind is a computing system is a matter of determining whether it has the relevant functional properties. (2007b, p. 113)

As we have seen, according to Piccinini's account of computation, the components of computational systems are functionally individuated. This means they are individuated by what they do (e.g. if a component turns external stimuli into computational vehicles, it is an input device). Computational vehicles, in turn, are also characterized by their functional properties (besides the appeal to the degrees of freedom of the system), and computations are defined by rules that are only sensitive to functionally relevant state variables that are medium-independent. In this sense, computational explanations that omit structural descriptions of systems seem to be a kind of functional analysis. Consequently, as I will argue in the following section, Piccinini's account of computational explanation appears to be in conflict with mechanistic criteria for an adequate explanation.

2.2 Computational Explanations as Mechanism Sketches

As we have seen in section 2.1, a computational explanation of a system is based on that system's functional properties. Following Piccinini and Craver's characterization of functional analysis, computational explanations will qualify as such. They characterize functional analysis as "the analysis of a capacity in terms of the functional properties of a system and their organization" (2011, p. 286). As Piccinini and Craver (2011) point out, functional analyses share important features with mechanistic explanations. Functional analyses often appeal to systems' internal states and explain the behavior of those systems in terms of processes over inputs and internal states that give certain outputs. In such cases, the internal states of systems are individuated functionally in terms of their relations to inputs, outputs, and other internal states. The mechanistic view holds that this type of analysis is a kind of mechanistic explanation, because the internal states they refer to are states of the components of a mechanism that can be appropriately identified by a complete mechanistic explanation.

Another type of functional analysis proceeds by individuating components of systems—based on their functional properties and without describing their structural properties, which is why Piccinini and Craver refer to them as 'black boxes'—, attributing processes or activities to them and attempting to explain the system's behavior in terms of these components and their activities. This type of analysis is also held by Piccinini and Craver to be a kind of mechanistic explanation, given that, despite the fact that it does not rely on structural properties of mechanisms, it does individuate components, their activities, and their organization. Insofar as functional analyses describe (although partially) a causally relevant aspect of a mechanism, they are, according to Piccinini and Craver, explanatory.[3]

[3] This is not to say that every functional analysis will count as a mechanistic ex-

In spite of their claim that a mechanism's components have both functional and structural properties and that explanations can emphasize either type of property depending on the context, Piccinini and Craver also claim that explanations which omit structural details, like functional analyses, are incomplete explanations, not full-blown mechanistic explanations. They state that:

> […] functional analyses are *sketches of mechanisms*, in which some structural aspects of a mechanistic explanation are omitted. Once the missing aspects are filled in, a functional analysis turns into a full-blown mechanistic explanation. (2011, p. 284)

They follow Craver (2006, 2007), who places mechanistic explanations on a continuum depending on how much detail they provide in the description of the mechanism. On one end of the continuum, Craver places *mechanism sketches*, which are abstract and incomplete descriptions of mechanisms. On the other end, he places *ideally complete descriptions*, which detail all the components and activities of the mechanism, as well as its organization. This continuum serves to establish a normative criterion to evaluate mechanistic explanations: the closer an explanation is to a sketch, the weaker its explanatory power. By this mechanistic criterion, computational explanations would be mere mechanism sketches, even under Piccinini's mechanistic account. As a consequence, they would not be good, full-blown mechanistic explanations, since these must detail both the structural and functional properties of components.

Following the dominant view on mechanistic explanation, the structural description of the components in physical terms is crucial to having a good explanation of a mechanism. As Craver claims, "Distinguishing good mechanistic explanations from bad requires that one distinguish real components from fictional posits" (2007, p. 131). Real components, as opposed to merely fictional ones, have a stable cluster of properties, are detectable with multiple techniques, are utilizable for the purposes of intervention, and are physi-

planation, but rather only those that reveal a relevant causal aspect of the underlying mechanism.

ologically plausible (Craver 2007). By Craver's standards, functional analyses with their typical 'box-and-arrow' diagrams are at best how-possibly models, since they usually include provisional terms (filler terms) and black boxes, and do not describe the components with enough structural detail as to establish whether they constitute real components or only fictional posits.

There seems to be some tension between the commitment to the mechanistic account for developing a proposal of computational explanation and the adoption of a kind of functional analysis for individuating a computing system (recall that whether a system is a computing system or not depends on whether it has the relevant functional properties). This tension stems from the fact that the mechanistic account regards functional analyses as incomplete explanations that need to be completed to attain explanatory power (Craver 2006, Piccinini & Craver 2011). Functional analyses can be considered explanatory to the extent that they describe mechanisms, but their explanatory power is very weak. As Piccinini and Craver (2011) maintain: "Mechanism sketches are incomplete because they leave out *crucial* details about how the mechanism works" (p. 292, emphasis added). By mechanistic standards, computational explanations as Piccinini reconstructs them leave out crucial details, and are thus explanatorily weak.

Of course, not every minute detail about the constitution of a mechanism is relevant to account for its behavior. It would be simply impossible (not only in practice, but also in principle) to describe all those details, and that is not what mechanists require.[4] Mechanistic criteria of adequacy require explanations to provide the details that are constitutively relevant to explain the phenomenon: "Good mechanistic explanatory texts describe all of the relevant components and their interactions, and they include none of the irrelevant components and interactions" (Craver 2007, p. 140). According to Craver (2007, ch. 4), a component is constitutively relevant to the behavior of a mechanism when there is a relation of mutual manipulability between that component and the mechanism as a whole, that is, when one can change the behavior of the whole by changing the behavior of the component, and vice versa. The criterion of constitutive relevance is intended to provide a way to distinguish between mere parts and components of a

[4] I am grateful to an anonymous referee for pointing this out.

mechanism, as well as between activities, properties, or organizational features that are relevant to the behavior of the mechanism and those that are not.

An adequate explanation should describe the *relevant* entities or components and how their organized activities account for the behavior of the system. The components and activities described should be the actual components and activities of the mechanism. That is, according to mechanistic criteria, it is not enough to provide an account of a system's functional organization or to functionally individuate components, since this does not allow us to determine whether they are the actual components of a system or not. Adequate explanations may omit some structural details that are not relevant for accounting for a system's behavior, but, as I understand mechanistic criteria, when almost all of them are missing, the explanation at hand is at best a mechanism sketch.

A reply to this concern could be that the account of computation developed by Piccinini, following the mechanistic view, incorporates a thesis of mutual restriction between structural and functional properties. Even if equivalent computational descriptions can correspond to different physical substrates, there will be certain structural features that those substrates must possess in order to be able to implement the computations. The relevant structural features are the degrees of freedom of the system (Piccinini & Craver 2011). As we have seen, a central feature of computational processes is their sensitivity to differences between portions of the vehicles. Thus, the physical substrates that implement the computations must exhibit enough dimensions of variation or degrees of freedom to instantiate different states. This is a structural constraint on computational mechanisms. Nonetheless, different mechanisms (with different structural properties) can have the same degrees of freedom. An explanation that appeals *only* to the degrees of freedom of a mechanism as a structural restriction on the functional properties it can instantiate is not an explanation that says a great deal about the mechanism's structural properties, and in this sense is still a sketch. Even if computational descriptions place constraints on the structures that can satisfy them—not every structure will have the right degrees of freedom and exhibit the right functional organization—, they still omit details that are needed for individuation of the relevant components.[5]

Piccinini's endorsement of the mechanistic account of explanation commits him to certain standards of adequacy that place computational explanations in the less explanatory end of the continuum of explanations. However, I believe that precisely those standards constitute one of the motivations for the adoption of the mechanistic account, as we will see. One of the main arguments Piccinini puts forward against previous accounts of computation intends to show that they do not offer an adequate distinction between systems that compute and those that do not. They either attribute computations to paradigmatic systems that do not compute (such as the planetary system or digestive system) or exclude paradigmatic computational cases (such as parsers and compilers) (Piccinini 2007b, 2012a, 2012b). The former is the case of mapping accounts of computation (e.g. Copeland 1996), and the latter is the case of the semantic view (Fodor 1975, Sprevak 2010).[6]

Mapping accounts explain computations in terms of mappings between physical states of a system and computational states, such that "the state transitions between the physical states mirror the state transitions between the computational states" (Piccinini 2012b, p. 226). Several versions of the mapping account incorporate different constraints to delimit which mappings are acceptable. However, as Piccinini points out, even with those constraints, all of them allow for establishing mappings between any physical system and some computational descriptions. In this sense, these accounts imply a form of pancomputationalism, which is the view that every system is a computing system. The strongest version of pancomputationalism has the consequence of trivializing computational explanations of cognition and of ordinary computers. Without further restrictions, simple mapping accounts entail acceptable mappings between almost any physical system

[5] Piccinini and Craver claim that computational explanations identify structural components. However, they seem to ground this claim on the fact that functional computational descriptions place constraints on the structures that can implement them, and constraining the properties of implementations is not the same as describing the structural properties of components.

[6] For a detailed reconstruction and discussion of several accounts of concrete computation, see Piccinini (2012a, 2012b).

and any computation. If any computation can be ascribed to a system, the explanation of its behavior in computational terms loses its power (Piccinini 2007b). The weaker forms of pancomputationalism also face a problem. If every system performs at least some computations, then mapping accounts do not provide an adequate distinction between systems that compute and systems that do not. As Piccinini (2009, p. 519) states, this "doesn't do justice to computer science, where only relatively few systems count as performing computations."

On the other hand, the semantic view aims to be more restrictive than the mapping account. According to this view, computational processes are defined over representations, and therefore the only systems that can be computational are those that manipulate representations. As a result, this view avoids pancomputationalism, since systems without representational content cannot be computational. However, Piccinini (2007b) argues that if having representational content is a necessary condition for performing computations, this view implies that paradigmatic computing systems such as parsers and compilers do not compute.[7]

In contrast to these accounts, the mechanistic view of computation aims to offer a proper distinction between systems that compute and systems that do not, as well as to provide the means to elaborate a taxonomy of computational systems based on their mechanistic properties (Piccinini 2007a, 2008a, 2012a). Craver's mechanistic criteria for an adequate explanation seem perfectly suited for the purpose of distinguishing actual computational systems from systems that are merely *described as* performing computations. As we have seen, the mechanistic view requires providing a structural and functional description of mechanisms in order to reveal their actual properties, as opposed to providing mere fictional descriptions (Craver 2006, 2007). I believe this is one of the motivations to reconstruct computational explanations as mechanistic.

According to the mechanistic view, no matter how useful fictional descriptions might be in specific contexts, they do not reveal the actual causal structure of mechanisms and are therefore explanatorily dubious. But the promise of the mechanistic view of distinguishing actual explanations

[7] Piccinini (2004, 2008b) offers further arguments against the semantic view.

from mere redescriptions comes at a cost. The identification of components in both functional and structural terms is at the core of the mechanistic approach, given that both types of properties contribute to individuate real components. Piccinini and Craver (2011) recognize that components have functional and structural properties, and that they do not need to be "neatly localizable." However, there seems to be an emphasis on structural properties when it comes to individuating *real* components, since a cluster of structural properties appears as the minimum requirement to identify a component: "[...] components need not be neatly localizable, visible or spatially contained within a well defined area. Any robustly detectable configuration of structural properties might count as a component" (p. 296, footnote). As a consequence, until explanations incorporate structural properties, they do not attain comprehensive explanatory power. However, Piccinini's account proposes focusing on the functional properties of the computing mechanisms, which falls far from this mechanistic normative criterion.[8]

Even though Piccinini and Craver recognize that for certain contexts, a sketch could be sufficiently explanatory, they still take sketches to be incomplete explanations. Moreover, they do not seem to take very seriously the scientific value of sketches, as the following quote illustrates:

> Sometimes a sketch provides just the right amount of explanatory information for a given context (classroom, courtroom, lab meeting, etc.). Furthermore, sketches are often useful guides to the future development of a mechanistic explanation. Yet there remains a sense in which mechanism sketches are incomplete or elliptical. (2011, p. 292)

Under the mechanistic account of computation, computations are defined in functional terms as manipulations of medium-independent vehicles. Computational explanations, in turn, are a kind of functional explanation

[8] There are other criteria of adequacy of explanations, such as describing the active, spatial, and temporal organization of a mechanism (Craver 2007), which computational explanations may satisfy. I am focusing here on the requirement of describing real components to argue that without an adequate mechanistic description of components, computational explanations will be incomplete.

since they only need to rely on the functional individuation of components and their activities and on the functional organization of those components. My concern is that, by mechanistic standards, those kinds of explanations are not good explanations. But, as Piccinini and Bahar (2012) recognize, computational explanations are often the best available for psychological phenomena. Thus, under this approach, the best current psychological explanations are held as mere mechanism sketches and thereby as bad explanations. This may not seem problematic for the mechanistic view since it holds computational explanations in psychology to be only partial and provisional explanations that should be eventually integrated into complete mechanistic explanations (Piccinini & Craver 2011). However, as Piccinini (2006, Piccinini & Bahar 2012) points out, explanations of neural processes appeal to computational descriptions, as well. Piccinini and Bahar defend that explanations of cognitive phenomena should take neural computation into account; that is, if those explanations make use of computations, they should be computations carried out by, or implemented in, neural processes. But given the way Piccinini proposes to account for computations, as manipulations of medium-independent vehicles, computational explanations of neural processes will still be highly abstract and sketchy, and therefore incomplete by mechanistic standards of evaluation.

As I have tried to show in this section, if computational explanations individuate computations and computational components in functional terms, they will be only partial and provisional explanations under the mechanistic view. If those explanations gain their place in the mechanistic continuum of explanations, it is only under the promise of a future structural completion. But, as I will argue in the next section, the structural completion of computational explanations does not necessarily make them explanatorily better. I will try to show that when computational explanations embody detailed structural properties, computations are no longer multiply realizable, and therefore computational explanations lose explanatory power.

3. The Individuation of Computations by Functional and Structural Properties

The commitment to the mechanistic criteria for an adequate explanation could suggest that computations and their vehicles should be individuated in terms of their functional *and* structural properties. In Piccinini's mechanistic account of computation, this appears to be the core of the proposal, i.e. to characterize all the computational components in both functional and structural terms. Following this proposal, processors, memory units, input devices, output devices, and computational vehicles should receive a structural characterization. As I pointed out in section 2, the main motivation behind this requirement seems to be the intention of being able to empirically discriminate between mechanisms that compute and mechanisms that do not. To be able to determine whether a particular physical system performs computations or not, the mechanistic account claims that it is necessary to have an actual description of the system.

Following the mechanistic account of explanation, functional analyses, which do not take into account the structural characteristics of the system described, are at best mechanism sketches waiting for a complete mechanistic description. As a result, it could be argued that the mechanistic account of computation should offer not only structural descriptions of the computational systems, but also a way of individuating computations and computational systems based on both functional and structural properties. The problem of adopting this alternative is that the multiple realizability of computational systems will no longer be possible. In section 3.1, I will present the notion of multiple realizability I am adopting and will show why a full-blown mechanistic description of computational systems is not compatible with their multiple realizability. In section 3.2, I will argue that the loss of multiple realizability is an undesirable consequence of an account of computation and computational explanation.

3.1 Full-blown Mechanistic Computational Explanations and Multiple Realizability

There is an ongoing debate about the notions of realization and multiple realization. The debate centers on the conditions for identification of cases of multiple realization. There is disagreement on the relata of the relation—whether they are properties, kinds, activities etc.—, the levels of the relation—whether the relata are at different levels of description (inter-level realization), or they can be at the same level (intra-level realization)—, and what constraints the relation is subject to. But there is some common ground, even if it is minimal:

> Occasionally it seems that the only common ground is that realization is a dependence relation that sometimes or always relates entities that figure in different explanatory schema, such as those of the special sciences and those of more fundamental sciences. (Polger 2010, p. 193)

I will follow Weiskopf (2011a) and maintain that realization is a relation between properties at different levels of organization. Multiple realizability occurs, at least, for functional kinds—i.e. kinds that are defined by some capacity, disposition or functional property—in cases in which several different properties at a lower level of organization exhibit such a capacity, disposition, or functional property.

This notion fits well with both the mechanistic account of computation and the issue of multiple realizability of computations for several reasons. First, Weiskopf makes use of mechanistic criteria to distinguish different realizers of functional kinds. He appeals to differences in the structural properties of the components, such as their size and location, and to the different possibilities of intervention and manipulation offered by each realizer. Second, Weiskopf's account focuses on properties or kinds described in the same way that Piccinini employs to characterize computational systems, that is, in terms of their functional properties. Weiskopf concentrates on the relation between functionally characterized kinds and the different mechanisms that can instantiate those kinds. Finally, Weiskopf accentuates precisely the common ground pointed out by Polger, since he proposes identifying kinds in terms of their roles in explanatory schemas (Weiskopf refers to scientific models). He thereby proposes to ground the notion of realization and the criteria for evaluating cases of multiple realizability in scientific practices.

By adopting this view, Weiskopf is able to respond to a line of criticism of the thesis that functional kinds are multiply realizable. Shapiro (2000, 2004) argues against the multiple realizability of functional properties, claiming that either there are no genuine cases of multiple realization (because many alleged cases of multiple realization pertain to realizers that are too similar in their causal structure to belong to different kinds) or, if there are, given that the realizers must belong to different kinds, they do not allow for interesting scientific generalizations. Hence, there is no interesting sense in which the realized kind counts as a scientific kind. In response to Shapiro, Weiskopf develops the following strategy. First, he gives examples of functional properties instantiated by different mechanisms that qualify as different kinds according to the sciences in charge of studying them. Second, he argues that those different mechanisms constitute a kind, given that they are useful in the relevant sciences by their role in a wide range of empirically validated models.[9]

Adopting this notion of multiple realizability, cases of multiple realization need to meet two conditions: systems with different structural properties must satisfy the same functional description, and that functional description must individuate a kind in virtue of being of explanatory use in the pertaining sciences. Cases of different mechanisms that satisfy the same computational description count as cases of multiple realization, provided that the computational description picks up a kind supported by empirically validated models. However, if computational explanations require individuation of computations and computational components both in detailed structural and functional terms, providing a complete mechanistic description, it will no longer be possible for systems with different structural properties to satisfy those descriptions. Once all the structural details are filled in, the result will be a description of a particular mechanism which implements a computational system, not a description of a computational system that is multiply

[9] For example, Weiskopf (2011a) analyzes the mechanisms that realize lateral inhibition –which produces the psychophysical phenomenon of Mach bands– in both camera eyes and compound eyes. Camera eyes and compound eyes differ in their range of cell types, the organization of cells, and the connectivity patterns among them, but both produce lateral inhibition processes and thus exhibit the Mach bands phenomenon. In this sense, they count as different realizers of a functionally characterized property (lateral inhibition), which in turn explains the presence of a scientifically interesting psychophysical effect (perception of Mach bands). For further examples of scientifically grounded cases of multiple realization, see Weiskopf (2011a) and Aizawa & Gillett (2009).

realizable.

It may be objected that structural properties are multiply realizable as well, and thus that a detailed structural description is still compatible with multiple realizability. Shape and size, paradigmatic mechanistic structural properties, are multiply realizable in different microstructures or different physical media.[10] A first answer to this objection comes from some considerations as to which cases count as multiple realizations. It can be argued that differences in constitution or microstructure are only multiple realizations in a trivial sense. As in Shapiro's (2000, 2004 ch. 2) corkscrew example, two waiters corkscrews made respectively of steel and aluminum do not count as genuine multiple realizations, because they do not differ in causally relevant ways. Differences in constitution are not always relevant for the causal explanation of the behavior of mechanisms. Accepting Shapiro's point, differences in constitution will not constitute multiple realization cases *per se*. This does not mean that there are no cases of multiple realization at all, nor that differences in constitution will never amount to multiple realization cases, but only that interesting cases will be the ones that exhibit, in mechanistic terms, differences that are relevant for explaining the behavior of the mechanism.[11] That is, to count as different realizers, two mechanisms need to produce the relevant phenomenon—in this case, computation—in different ways.

A second answer points out that, disregarding whether the microstructural differences amount to multiple realizers or not, if computations—or computational systems—are described in detailed structural terms, then it is not possible to realize them in different structures. A complete description of structural properties and organization might leave open the possibility of implementations in different media, but it precludes the possibility of implementations in mechanisms with different structural components. It is

[10] I am grateful to two anonymous referees for pointing this out.

[11] It is interesting to note, however, that concerning how to individuate kinds of mechanisms, Craver (2009, p. 586) maintains: "[...] it will not suffice to demand that the differences in underlying mechanisms must make a causal difference to (or be otherwise explanatorily relevant to) the behavior of a mechanism as a whole because any detectable difference in the underlying mechanism must make some such causal difference."

in this sense that I take complete structural descriptions to be incompatible with multiple realizability.

3.2 Two Problematic Consequences

The incompatibility with the multiple realizability of computations and computational systems would be an undesirable consequence for an account of computations and computational explanations for two reasons. First, it would fail to reflect some ways of understanding computations in computer sciences. Second, and more importantly, if they are incompatible with multiple realizability, in some cases computational explanations will be of little use in discriminating between systems that compute and those that do not, which is one of the aims of Piccinini's account. In what follows, I will address these consequences in turn.

First, leaving aside the possibility of the multiple realization of psychological properties, paradigmatic computational systems are widely taken to be multiply realizable. The mechanistic account intends to do "justice to the practices of computer scientists and computability theorists" (Piccinini 2007a, p. 501). Those practices include the functional individuation of computations and the assumption that a given computational system can be implemented in different physical media. For example, in Turing's (1950, p. 439) terms:

> The fact that Babbage's Analytical Engine was to be entirely mechanical will help us to rid ourselves of a superstition. Importance is often attached to the fact that modern digital computers are electrical, and that the nervous system is also electrical. Since Babbage's machine was not electrical, and since all digital computers are in a sense equivalent, we see that this use of electricity cannot be of theoretical importance. […] In the nervous system chemical phenomena are at least as important as electrical. The feature of electricity is thus seen to be only a very superficial similarity. If we wish to find such similarities we should look rather for mathematical analogies of function.

According to this traditional interpretation, different mechanisms –which could have radically different physical properties– can realize the same computational system when they are able to execute the same set of functions.

For example, our ordinary computers implement logic gates (which are primitive computing components) in electronic circuits, but there are also implementations of logic gates on dominoes (O'Keefe 2007, Stevens 2012). These different physical structures can satisfy the same (functional) computational description. A simple electronic circuit and a domino arrangement will qualify as genuine different realizers of an AND gate, for example, because they differ in many mechanistic properties. In different media, the components of a computational system will differ in, e.g., size, shape, location, and orientation, which are structural individuating properties of mechanisms. To state this in Shapiro's terms, an electronic AND gate and a domino gate count as different realizers, because they perform the same computation in different *mechanistic* ways. If their properties are functionally described, they are not different in any relevant way (if they were functionally different, they would not count as realizers of the *same* kind). But such a description would be made of filler terms, such as 'receiving inputs and yielding an output.' In contrast, a description that takes their structural properties into account will describe their activities as 'conducting electricity' in the electronic gate, and 'propagating a wave front' or 'colliding' in the domino gate.

If one takes seriously the requirement of an adequate individuation of real components and their activities, then electronic circuits and domino arrangements are different realizations. I consider the example of logic gates of interest, because if these primitive computing components are multiply realizable, then the computing systems built out of combining them (plus other components) will be multiply realizable, as well (provided that the realizers of different components are compatible in the sense of being able to operate together). Different mechanisms can implement the same set of computations. This is the way computations are understood in at least some branches of computer sciences. In fact, there is an area of research within computer sciences dedicated to implementing computational systems in unconventional media, such as DNA molecules or enzymes.[12] To reflect

[12] Some recent developments in this area of research can be found in Calude *et*

this use of computation in computer sciences, an adequate account of computation should allow multiple realizability. To accomplish this, computations cannot be defined taking many structural details into account.

Turning to the second and more important consequence, it could be possible to give a very detailed description including all the physical characteristics of a computational system. However, such a description would miss the mark in providing what is necessary to establish whether a system is computational or not. To explain why this is so, I will turn to an application of the mechanistic account of computation. As I pointed out at the beginning of this paper, one of Piccinini's aims is to develop a clear account of concrete computation that can contribute to evaluate computational explanations, especially computational explanations of psychological capacities and computational theories of mind. Piccinini and Bahar (2012) put the mechanistic account of computation to work in order to establish whether the nervous system is a computational system. They argue that the neural processes that explain cognition are computations in the generic sense, hence the nervous system is a computational system in the generic sense. They put forward two different arguments to support that claim; one based on the nervous system's functional organization and another based on the fact that it processes information. The first one runs as follows:

1. The neural processes that explain cognition manipulate medium-independent vehicles.
2. Manipulations of medium-independent vehicles are computations in the generic sense.
3. Therefore, the neural processes that explain cognition arecomputations in the generic sense. (Piccinini & Bahar 2012, p. 10)

In support of the first premise, they affirm that spike trains are the primary vehicles of neural processes and refer to evidence that suggests that neural processes depend on dynamical aspects of those vehicles, such as spike rates and spike timing. To maintain that those properties or aspects are independent of the neural medium, they give two reasons, both of which I find

al. (2011).

problematic in different senses.

The first is that these dynamic aspects are independent from the physical properties of the different stimuli; that is, the nervous system processes spike trains regardless of whether the stimulus is, e.g., auditory or visual. I do not believe this amounts to medium-independence of the vehicles, but rather to some sort of vehicle homogeneity throughout the system. Still, the fact that the vehicles are homogeneous independently of the stimulus does not imply that those vehicles are medium-independent; instead, the system could have input devices, the same kind of components Piccinini (2007a) describes, which have the function of turning external stimuli into computational vehicles. I take the homogeneity throughout the nervous system of the dynamical parameters that are functionally relevant for neural processes as evidence in favor of such input devices. But those dynamical properties could still be highly dependent on the physical properties of the medium. Their homogeneity across the system does not show their medium independence. This is why I believe that the strongest reason in favor of the medium-independence of the vehicles of neural processes is the second: the possibility of implementing those dynamical and functional properties in different physical media, which entails a case of multiple realizability. I address this second reason and its problems below.

The strongest support for the claim that neural processes manipulate medium-independent vehicles comes from the fact that the dynamical properties on which those processes depend can be implemented in several physical media. Piccinini and Bahar mention silicon-based circuits. This implies that it is necessary to have a characterization of computations that is highly abstract, one that specifies the functions being computed without appealing to the neural substrate in such a way that it is possible to test whether that same function could be implemented on a different basis. It makes no sense to discuss the implementation of a neural system in another substrate since it is constitutive of a neural system to be instantiated in neurons. However, if there is another description available, one that dispenses with the physical details, it makes sense to evaluate whether that description could be satisfied by different implementations. Perhaps requiring evidence of multiple implementations is too strong a criterion for establishing that a system computes, but in order to suggest that something could have different implementations or realizations, it is necessary for that something to be described in a medium-independent way. Such a description is what computational explanations provide, and this is why Piccinini's account of computations seems to be, as we have seen in section 2, a functional analysis.

It might be objected, following Shapiro (2000, 2004, ch. 2) that neural systems and silicon-based circuits are multiple realizations of a single computational system only in a trivial sense, since they exhibit the same electrical properties and the same organization, which are the relevant properties to account for the behavior of the system. However, as Weiskopf (2011a) points out, mentioning the example of silicon-based artificial visual systems, artificial and neural systems exhibit different components, an example of which is that artificial systems dispense with neurotransmitters. The fact that the same behavior can be accomplished by a different system without neurotransmitters, does not show that they are irrelevant to the behavior of the original system, but only that this particular phenomenon can be realized in mechanisms with different components.

As I have mentioned, Piccinini and Bahar develop a second argument to show, without relying on the multiple realizability of computations, that the nervous system computes in the generic sense. If this second strategy is enough to argue for generic computation, then my objection against the structural individuation of computations would be undermined. Piccinini and Bahar (2012, p. 11) maintain that "some neural processes that explain cognition process semantic information," and given that the processing of semantic information entails computation in the generic sense, those processes are therefore computations in the generic sense. This argument does not depend on the multiple realizability of the computations performed by the nervous system. However, this second argument will not apply to every case of computation because, as Piccinini and Bahar defend, not every computing system processes semantic information. The processing of semantic information entails computation in the generic sense, but the converse relation does not hold. Some systems compute without processing semantic information, for example "a computer programmed to manipulate meaningless digits according to some arbitrary rule" (Piccinini & Bahar 2012, p. 5). Thus, this second type of argument will not be available to support that a system computes for every case, but only for cases of information-processing systems.[13]

It seems that from this application of Piccinini's account and the two arguments for the fact that the nervous system is a computational system in the generic sense, multiple realizability is, in fact, a strong reason in support of computation. This seems to be a paradoxical result, given that, as I am arguing, the mechanistic view emphasizes the need for detailed structural accounts, which do not seem compatible with multiple realization. I want to stress that I am not arguing against the possibility of giving a complete description of a computational system, one which includes detailed structural properties of its implementation, although that could also prove problematic. Weiskopf (2011b) elaborates an argument in that direction for the case of computational explanations of cognitive capacities. He questions the possibility of obtaining a detailed description of the neural basis of psychological capacities. Moreover, he argues that even if we had some description available, structural and functional descriptions could differ:

> Even if brains were less than staggeringly complex, it would still be an open question whether the organization that one discovers in the brain is the same as the one that structures the mind, and vice versa. (2011b, p. 328)

Weiskopf examines several cognitive models and claims that they do not posit components that have a straightforward correspondence to the physi-

[13] Another possibility would be to try to determine whether a system is computational by appealing to the different species of computation, without appealing to multiple realizability. That is, perhaps it is possible to identify a system as, e.g., a digital computing system, if it is possible to determine that it processes strings of digits. However, this possibility would only allow one to test for well-known kinds of computations, such as digital. If, as Piccinini and Bahar (2012) suggest, there are other kinds of computation not yet precisely characterized, such as the one performed by the nervous system, it will not be possible to test whether any system performs them. Although it is possible to establish that the nervous system computes in the generic sense, it is not yet possible to characterize the kind of computations it performs (Piccinini & Bahar 2012). Therefore, maybe for some systems it may be possible to determine that they compute because they operate, e.g., over strings of digits, but this will not be possible for every computing system.

ological organization of the brain. According to Weiskopf, some computational explanations of cognitive capacities describe systems that do not have, and cannot have, a complete mechanistic description.

The problem I am pointing out is a different one. I maintain that, even in cases in which a complete mechanistic description is possible, not only should it not be a requisite to constitute a good computational explanation, but, moreover, it would not be in virtue of that detailed description that we are able to determine whether that system implements computations or not. This latter point was one of the main goals of adopting the mechanistic view to develop an account of computation. If computational explanations incorporate too many structural details, as normative mechanistic criteria demand, not only will they undermine the possibility of multiple realizability, but, as I have tried to show by discussing an application of the mechanistic account, they will also lose some explanatory advantages. As Piccinini points out, computational descriptions remain abstract from physical details of the computational vehicles in order to account for input-output mappings. The demand of giving every explanation in structural terms would obscure the possibility of properly accounting for computations, because it would collapse the characterization of computations with that of their implementations. The characterization of computation given by Piccinini tries to avoid this undesirable consequence and omits the structural details. But, as I argued in section 2, it renders computational explanations weak according to mechanistic criteria.

4. Conclusion

The mechanistic account of computation characterizes computations and computational systems in terms of the mechanistic properties of systems. As we have seen, it does so by focusing almost completely on the functional properties of mechanisms, which leads to the dilemma I have presented. Either computational explanations emphasize the functional aspects of a mechanism, respecting a common use of computation in computer sciences, but falling on the less explanatory end of the continuum of mechanistic explanations, or they emphasize the structural aspects of mechanisms, thereby giving up the multiple realizability of computational systems and, as a result, losing a central feature of the notion of computation.

To embrace the first option, that is, the functional individuation of computations and computational vehicles, the advocates of the mechanistic view of explanation should loosen up the normative criteria of adequacy of explanations. However, this does not fit with mechanistic goals since, according to the mechanistic view, the individuation of components by their structural properties plays a crucial role in distinguishing mere useful fictions from descriptions of actual mechanisms. Therefore, it does not seem plausible that the mechanistic view will relax the normative criteria. To embrace the second option, the structural individuation of computations and computational vehicles, seems a better alternative for the mechanistic approach. However, as I have argued, this option is not compatible with the view that computations and computational systems are multiply realizable, which is not only a common view in computer sciences, but is also used as evidence of cases of computation. As I have aimed to show, this leaves the mechanistic account of computation in a difficult position. Given the difficulties that the mechanistic approach faces in providing an adequate notion of computation, and since this notion is central to explanations in cognitive sciences, this further reveals the limits of the mechanistic approach of computation in accounting for explanations in cognitive sciences.

References

Aizawa, K. & Gillett C. 2009. The (multiple) realization of psychological and other properties in the sciences. Mind & Language 24, 181-208.

Bechtel, W. 2008. Mental Mechanisms: Philosophical Perspectives on Cognitive Neuroscience. London: Routledge.

Copeland, B. J. 1996. What is computation?. Synthese 108, 335-359.

C. S. Calude, J. Kari, I. Petre & G. Rozenberg (Eds.) 2011. Unconventional Computation. Berlin: Springer.

Craver, C. 2006. When mechanistic models explain. Synthese 153, 355-376.

Craver, C. 2007. Explaining the Brain. Oxford: Clarendon Press.

Craver, C. 2009. Mechanisms and natural kinds. Philosophical Psychology 22, 575-594.

Fodor, J. A. 1975. The Language of Thought. New York: Thomas Y. Crowell.

Machamer, P., Darden, L. & Craver, C. 2000. Thinking about mechanisms. Philosophy of Science 67, 1-25.

O'Keefe, S. 2007. Implementation of logical operations on a domino substrate. In A. Adamatzky, B. De Lacy Costello, L. Bull, S. Stepney and C. Teuscher (Eds.), Unconventional Computing, 20-29. Luniver Press.

Piccinini, G. 2004. Functionalism, computationalism, and mental contents. Canadian Journal of Philosophy 34, 375-410.

Piccinini, G. 2006. Computational explanation in neuroscience. Synthese 153, 343-353.

Piccinini, G. 2007a. Computing mechanisms. Philosophy of Science 74, 501-526.

Piccinini, G. 2007b. Computational modelling vs. Computational explanation: Is everything a Turing Machine, and does it matter to the Philosophy of Mind? Australasian Journal of Philosophy 85, 93-115.

Piccinini, G. 2008a. Computers. Pacific Philosophical Quarterly 89, 32-73.

Piccinini, G. 2008b. Computation without representation. Philosophical Studies 137, 205-241.

Piccinini, G. 2009. Computationalism in the philosophy of mind. Philosophy Compass 4, 515-532.

Piccinini, G. 2010. The mind as neural software? Understanding functionalism, computationalism, and computational functionalism. Philosophy and Phenomenological Research 81, 269-311.

Piccinini, G. 2012a. Computation in physical systems. The Stanford Encyclopedia of Philosophy (Fall 2012 Edition), E. N. Zalta (ed.). Retrieved on September 2012, from the website: http://plato.stanford.edu/archives/fall2012/entries/computation-physicalsystems/.

Piccinini, G. 2012b. Computationalism. In E. Margolis, R. Samuels & S. Stich (Eds.) The Oxford Hanbook of Philosophy of Cognitive Science, 222-249. Oxford: Oxford University Press.

Piccinini, G. & Bahar, S. 2012. Neural Computation and the Computational Theory of Cognition. Cognitive Science, 1-36.

Piccinini, G. & Craver, C. 2011. Integrating psychology and neuroscience: functional analyses as mechanism sketches. Synthese 183, 283-311.

Piccinini, G. & Scarantino, A. 2011. Information processing, computation, and cognition. Journal of Biological Physics 37, 1-38.

Polger, T. W. 2010. Mechanisms and explanatory realization relations. Synthese 177, 193-212.

Shapiro, L. 2000. Multiple realizations. The Journal of Philosophy 97, 635-654.

Shapiro, L. 2004. The Mind Incarnate. Cambridge: MIT Press.

Sprevak, M. 2010. Computation, individuation, and the received view on representation. Studies in History and Philosophy of Science 41, 260-270.

Stevens, W. M. 2012. Computing with planar toppling domino arrangements. Natural Computing 11, 665-672.

Turing, A. M. 1936. On Computable Numbers, with an Application to the Entscheidungsproblem. Proceedings of the London Mathematical Society Series 2 42, 230-265.

Turing, A. M. 1950. Computing machinery and intelligence. Mind 59, 433-460.

Weiskopf, D. A. 2011a. The functional unity of special science kinds. British Journal for the Philosophy of Science 62, 233-258.

Weiskopf, D. A. 2011b. Models and mechanisms in psychological explanation. Synthese 183, 313-338.

Manuscript received: April 7, 2013, in revised form: May 12, 2013