Computational Systems as Higher-order Mechanisms

Jorge I. Fuentes

Please cite from https://link.springer.com/article/10.1007/s11229-023-04482-y

Abstract

I argue that there are different orders of mechanisms with different constitutive relevance and individuation conditions. In common first-order mechanistic explanations, constitutive relevance norms are captured by the matched-interlevel-experiments condition (Craver et al. 2021). Regarding individuation, we say that any two mechanisms are of the same type when they have the same concrete components performing the same activities in the same arrangement. By contrast, in higher-order mechanistic explanations, we formulate the decompositions in terms of generalized basic components (GBCs). These GBCs (e.g., logic gates) possess causal properties that are common to a set of physical systems. Mechanistic explanations formulated in terms of GBCs embody the epistemic value of horizontal integration, which aims to explain as many phenomena as possible with a minimal amount of abstract components (Wajnerman Paz, 2017a). Two higher-order mechanisms are of the same type when they share all the same GBCs, and they are organized as performing the same activities with the same interactions. I use this notion of mechanistic order to enhance the mechanistic account of computation (MAC) and provide an account of the epistemic norms of computational explanation and the nature of medium-independent functional properties and mechanisms. Finally, I use these new conceptual tools to address four criticisms of the MAC.

Keywords: Mechanistic account of computation, mechanisms, computation, implementation, medium-independence.

1. Introduction

The mechanistic account of computation (MAC), as detailed by Piccinini (2015, 2020), is a prominent approach to determining what computation is in computer science and computational cognitive neuroscience.[1] MAC uses the term "mechanism" as in the new mechanistic philosophy of science (e.g., Bechtel, 2009; Craver, 2007; Glennan, 2017). More specifically, neomechanists emphasize that some special sciences, such as some areas of biology and cognitive neuroscience, employ mechanisms to explain phenomena instead of laws, functional analysis, or other kinds of explanation. Mechanisms are systems in which the parts interact to produce or constitute an explanandum phenomenon.

To fully characterize the notion of computation, the MAC employs three key concepts: medium independence, teleological function, and rule. Computational mechanisms are said to be medium-independent, meaning, among other things, that computational mechanistic explanations do not appeal to the medium in which the computation is implemented beyond its degrees of freedom. By contrast, they include mathematical and logical decompositions of the function computed by the mechanism in subfunctions computed by subcomponents. One of the goals of the MAC is to determine when a concrete medium-dependent mechanism is implementing some computation in an observer-independent way.

MAC has received multiple criticisms over the years, which can be grouped into four kinds: skeptics about abstract mechanisms (SAs), skeptics about medium-independent teleological functions (SFs), skeptics about medium-independent vehicles (SVs), skeptics about the viabiliy of mechanistic criteria for the individuation of computation (SIs) (see section 2).

[1] MAC has many versions depending on who articulates it (e.g., Coelho Mollo, 2018, 2019; Fresco, 2014; Kaplan, 2011; Kuokkanen, 2022a, 2022b; Milkowski, 2013; Piccinini, 2015, 2020). Here, I focus mainly on Piccinini's version.

I will argue that, to fully understand what a medium-independent mechanism is, we need to apply both Kuokkanen's (2022b) distinction between vertical and horizontal abstraction (see section 3) and the distinction I shall draw between regular (first-order) mechanistic explanation and higher-order mechanistic explanation. This concept of order is heir to Kim's (1998) concept but adapted to the mechanistic norms of explanation.

Higher-order mechanistic explanation has a different set of individuation conditions and additional constitutive relevance conditions (see section 4). The epistemic value behind the distinctive conditions of higher-order mechanistic explanation is horizontal integration (Wajnerman Paz, 2017a). As a value, horizontal integration establishes that in abstract mechanistic explanations, we should aim to explain as many phenomena as possible with a minimal amount of basic components[2]. Based on this value, I formulate an epistemic norm –namely, GBC– that governs the kinds of components in terms of which higher-order mechanistic explanations should be formulated.

Note that, regarding neural computation, higher-order mechanistic explanations are different from functional analysis explanations in classical cognitive sciences in the following ways. First, they are not discovered independently or autonomously from neuroscience since they are abstracted away from medium-dependent mechanistic explanations. These medium-dependent explanations, in turn, are obtained by performing experiments on concrete systems in the way described by Craver et al. (2021). This ensures that they are *how-actually* models and not simply *how-possibly* ones (Piccinini & Craver, 2011), which means that every computational sub-process included in the decomposition must match a sub-process performed by any implementation of that computational mechanism (see also Kaplan, 2011). Finally, higher-order mechanistic explanations are constitutive in that they appeal to

[2] Wajnerman Paz (2017a) explains: "... the kind of unification that I am considering in this section occurs when different phenomena can be explained by *different* models that describe sets of *shared* or *common* basic components or operations". I use this idea to formulate an explicit constitutive relevance condition for abstract mechanisms.

sub-activities *of parts*. This is not always true of functional analysis, which can decompose a phenomenon into sub-activities *of the whole*. For instance, when cooking something, every step is (supposedly) carried out by the person as a whole, so this analysis is functional, in the classical sense, but not mechanistic in the standard sense.

Although Craver et al.'s matched-interlevel experiments condition (MIE) summarizes medium-dependent mechanistic explanations' constitutive relevance well, it does not give us an epistemic guide as to what details we should omit when horizontally abstracting in order to obtain medium-independent mechanistic explanations. I will propose a generalized basic components (GBC)[3] condition based on the epistemic goal of horizontal integration mentioned above to complement MIE in higher-order mechanistic explanations.

Furthermore, as a consequence of the introduction of the GBC condition, I will propose a distinction between medium-independent functional properties and medium-independent mechanisms. That distinction will allow me to characterize the relationship between orders and Marr's (1982/2010) levels of analysis. Marr defined three levels of analysis to study neurocognitive phenomena, namely, the implementation, the algorithmic, and the computational. The implementation level concerns the neurobiology of cognitive operations. The algorithmic concerns the computational strategies employed by the target system of a psychological explanation. Finally, the computational level explains the relationship between the inputs and outputs of a cognitive operation without containing details about the implementation or the strategy employed. I will argue that both what Marr calls "algorithmic" and "computational" levels are computational but in different senses. On the one hand, only the former satisfies the constitutive relevance condition I propose for higher-order mechanistic explanations. These explanations need to be formulated in terms of generalized basic components common to a dominion of explanation – e.g., digital

[3] I will call GBC the criterion and GBCs (plural form) the abstract components used to formulate an abstract mechanistic explanation. Context should make clear which one is being referred to in every case.

computations can be decomposed onto logic gates and their relationships. On the other hand, the latter only describes a medium-independent computational teleofunctional property.

The distinction between the two kinds of mechanistic abstraction and the notion of higher-order mechanistic explanation will allow us to reconsider the minimal conditions for a mechanism to implement computations (section 5). In the case of GBCs, which cannot be further mechanistically decomposed to obtain more basic computational components, we will study the conditions they need to satisfy to be appropriately considered computational.

Finally, I will demonstrate the fruitfulness of the current proposal by using it to address four groups of criticisms against the MAC that have been accumulating over the years (see section 2 for the criticisms and section 6 for the responses).

2. The mechanistic account of computation and its critics

In the current section, I aim to summarize the discussion surrounding the MAC throughout these years. I start by providing a synoptic view of its main postulates as well as what distinguishes it from other accounts of physical computation. Next, I analyze the criticisms that this account has received through time and classify them into four groups. Each group targets different aspects of the view, namely, its distinction between sketches and explanations, the viability of the notion of medium-independent teleofunctions, the viability of the distinction between medium-independent vehicles and medium-dependent ones, and the ability of the account to provide straightforward criteria of computational individuation. In section 6, after proposing an enrichment of the MAC via defining higher-order mechanisms, I will address each of the mentioned criticisms.

According to the mechanistic account of computation, as articulated by Piccinini (2020), a computing system is a medium-independent functional mechanism whose teleological function is to process vehicles according to a rule. As said above, mechanisms are systems in which the parts interact causally to produce or constitute a phenomenon. That the mechanism is medium-independent means that it is multiply realizable and the vehicles it processes are also multiply realizable.

Piccinini provides the following examples. A mousetrap is a multiply realizable mechanism in the sense that it can be constructed out of springs, a laser detector that drops a cage on the mouse, a hole covered with a piece of fabric with cheese in the center, etcetera. The relevant feature, in this case, is the state transition from 'free mouse' to 'caught mouse'. Nevertheless, the medium over which the mechanism acts, i.e., the mouse, is fixed. So, there is no multiple realizability of the vehicles, i.e., the object that is acted upon, and, in turn, no medium independence. In contrast, an exponentiator that takes two values, X and Y, and returns X^Y is a mechanism that is both multiply realizable and whose vehicles could be made out of any suitable material with the resources, or degrees of freedom, to represent the required values. So, the exponentiator is a medium-independent mechanism.

Additionally, in this account, that the mechanism is functional means that the mechanism has a normative function that it can fulfill well or poorly (see also Garson, 2013). We must be able to assess that degree of fulfillment objectively. In the case of computation, this amounts to the fact that we can assess whether a system has computed correctly or miscomputed. The definition of teleological function (or teleofunction, for short) that supports the MAC (Piccinini, 2020, p. 76) is the following:

Tokens of type X have function F if and only if F is a causal role and performing F by tokens of X provides a regular contribution to a goal of organisms.

For example, the heart pumps blood all over our bodies. To do so is one of its causal roles; other causal roles include producing noise and electrical signals that allow us to measure its activity on an electrocardiogram. Of these three causal roles, only pumping blood helps us survive (without medical interventions). Since survival is one of the organism's goals, the heart's teleological function is pumping blood so as to bring nutrients and oxygen to every cell in the body. Poor pumping would lead to difficulties in the capacity to survive. In this case, we would say that the heart malfunctions.

The main advantages of the MAC, as opposed to other accounts of computations such as the semantic (e.g., Shagrir, 2022), simple mapping (e.g., Searle, 1992), causal mapping (e.g., Chalmers, 1996), step satisfaction (Cummins, 1989), etcetera, are three. First, it allows us to account for miscomputing as a case of malfunctioning. Second, it allows us to rule out non-functional systems, such as planetary systems or digestion, as candidates for being computers. Third, it allows us to define a generic type of computation in which digital, analog, or neural computation are subgenera.

Ruling out non-functional systems from counting as computers is important to avoid limited pancomputationalism. Limited pancomputationalism is the thesis that every physical system computes at least one function. Its plausibility arises from the fact that literally every physical system could be mathematically modeled at least in one way. Since that mathematical model could be considered as the computation that the system is carrying, pancomputationalism is supposed to follow. However, to avoid this conclusion, MAC introduces the difference between computational modeling and computational implementation and establishes that only certain kinds of teleofunctional systems implement computations, although every physical system can be computationally modeled.

MAC has received several criticisms over the years that appear to threaten it as a viable approach to computation. The critics can be grouped into four categories: skeptics about abstract mechanisms (SAs), skeptics about medium-independent teleological functions (SFs), skeptics about medium-independent vehicles (SVs), and skeptics about the viabiliy of mechanistic criteria for the individuation of computation (SIs). Let us say something about each in turn.

SAs usually employ the distinction between mechanism sketches and mechanistic explanations (Piccinini & Craver, 2011) to say that computational explanations should always be considered sketches by mechanists. Since sketches need to be fleshed out to become full-blown mechanistic explanations, this kind of skeptic claims that the mechanist is forced to diminish the value of computational explanations both in cognitive neuroscience and in computer science. Haimovici (2013) claims that MAC supporter confronts a dilemma. Either she ends up with a mere sketch of a mechanism, or by having a full-blown explanation, it loses the characteristic multiply realizability of computation. Chirimuuta (2014) claims that the introduction of the notion of mechanism sketches necessarily implies that the mechanist is committed to a principle she calls 'the more details, the better' (MDB). This principle establishes that the mechanist should strive to achieve explanations as physically detailed as possible. According to her interpretation, if the mechanist leaves any detail out, she has a sketch, not a full explanation. Also, Coelho Mollo (2018) claims that since computational explanations only restricts the degrees of freedom of a system, they cannot be considered proper mechanistic explanations. In other words, they pose too few constraints to be regarded as mechanistic, properly speaking. Thus, claims Coelho Mollo, mechanistic explanations can be provided only at Marr's implementation level. All these critics share the idea that, for an explanation to be mechanistic, it requires more details than a computational explanation. Additionally, the usual mechanistic condition for constitutive relevance cannot be used in isolation in abstract mechanisms. If we try to use the Mutual Manipulability (MM) condition or its more recent version, matched interlevel experiments (MIE), we must perform experiments on specific implementations. But that includes medium-dependent properties of that specific implementation that are not present in others (more on this in section 4).

SFs (e.g., Coelho Mollo 2019) claim that only mechanisms described with their implementation details could be ascribed a teleological function. This prevents abstract computational structures from being functional[4]. Since having a teleological function to compute is crucial to MAC, SF implies that MAC is not viable.

SVs deny that mechanisms that possess medium-independent vehicles can be separated from those that do not in a principled way. Maley (forthcoming) claims that, if the criterion is to be only sensitive to specific dimensions of variation, then systems such as a cylinder lock would count as computational, given that, for instance, cylinder locks are only sensitive to the geometrical patterns corresponding to the shape of keys. By contrast, if the criterion is to follow certain rules that are already specified as medium-independent, then neural computation would not be possible as neural processes do not appear to follow such rules. For that reason, Maley claims that the whole project of the MAC is condemned to failure unless coupled with semantic constraints.

Finally, SIs (e.g., Coelho Mollo, 2018; Shagrir, 2022) claim that, while the mechanistic norms allow us to individuate implementation mechanisms, they are not useful to single out computations performed by such mechanisms. In that regard, Shagrir states that to individuate computational properties, it is necessary to use semantic criteria. Coelho Mollo (ibid.) formulates a version of the MAC – which he calls amended MAC – in which mechanisms are conceived as only individuating implementations. By contrast, computations are individuated functionally. The distinction he draws between functional individuation and mechanistic implementation is based on the same intuitions that guide SAs, namely, that mechanistic abstraction must be limited for a system still to be called a mechanism properly. Coelho Mollo claims that restrictions on the degrees of freedom are insufficient for that job.

Before moving on to the next section, it is important to mention, regarding the latter two groups of critics, that both Maley (forthcoming) and Shagrir (2022) present their

[4] Coelho Mollo provides his own solution to this problem. However, his solution is based on assuming a distinction between mechanistic implementation and functional individuation, which entails that there are no higher-order mechanisms in the sense defined in this article. I will reject this solution since, as I will argue, abstract mechanisms are genuine mechanisms, and I will offer an alternative solution.

skeptical arguments to show that semantic properties are essential to computation. To clarify things, both mechanists and semanticists about computation claim that information processing entails computation and that neurocognitive systems process information and, as a consequence, perform computations. The difference is that semanticists also maintain the conditional in the other direction –i.e., computation entails information processing– making semantic properties essential to computation. I do not have the space here to fully do justice to semantic approaches to computation. However, I do argue, in section 6, that at least the mentioned skeptical arguments –namely, SVs and SIs– do not succeed.

3. Vertical and horizontal abstraction

In this section, I am going to present Kuokkanen's (2022b) distinction between horizontal and vertical abstraction in mechanistic explanations and argue for its relevance to MAC. In the following section, I will use this distinction to help define higher-order mechanisms and what it means to say they are abstract.

Mechanisms are multilevel hierarchies of organization that we find in many sciences, especially in cognitive neuroscience and other biological sciences. In mechanistic explanation, we explain a functional property of a system by decomposing it into its constituent parts (which in most cases are sub-mechanisms), the activities that those parts perform, and the relationships between them. Since sub-mechanisms are, in turn, susceptible to mechanistic explanations, in principle, we can keep going until no mechanistic functional units are to be found (e.g., elementary particles). Still, this is not how science usually works: the lower the mechanistic level, the more details it includes, but lower-level details are often irrelevant to explaining the phenomenon of interest, too many to include, or unknown (e.g., Boone and Piccinini 2016). Because of this, mechanistic explanation requires abstracting away from lower-level details. For example, when explaining a computational coding strategy of some neuronal population, we do not go all the way to how ion channels in the neural membrane work. Instead, we focus on the mechanistic level immediately below the system under study and abstract away from all the levels below. Kuokkanen calls this kind of abstraction *vertical*.

In addition, once the relevant mechanistic level(s) has been fixed, we need to choose the appropriate amount of mechanistic detail with which to describe them. Kuokkanen defines *horizontal* abstraction as a descriptive abstraction[5], i.e., an omission of detail, within a fixed mechanistic level of description (cf. Boone & Piccinini, 2016). For example, a connectome depicts only directed topological relations between neurons. Any details about their constitution, geometrical shape, size, etcetera, are left out. Even so, for some functional explanations, a connectome is all we need (see, e.g., Levy & Bechtel, 2013).

Furthermore, in neurocomputational explanations, all that is depicted are abstract relationships for processing inputs into outputs. No detail about the specific biological structures is needed, and they are even considered harmful if they decrease the generality of an explanation. The reason is that computational cognitive neuroscience aims for what Wajnerman Paz (2017a) calls 'horizontal integration'. Specifically, it aims for integration at "the same degree of abstraction that allows us to explain different capacities employing (the description of) a common set of basic operations" (p. 17). I will explore the epistemic importance of this kind of integration in the next section.

Since the relevant kind of omission here occurs within fixed levels of a mechanistic hierarchy, it is a horizontal abstraction. Moreover, since this is the kind of abstraction that

[5] Kuokkanen speaks of descriptive abstraction and claims the MAC does not entail an ontological commitment to platonic abstract objects. While I agree with the latter, I disagree that a merely descriptive view of abstraction is needed to avoid such a commitment. We are presented here with a false dilemma. Consider the following example. When describing something as a bacterium, many details about it are omitted and, therefore, to say "X is a bacterium" is, in this sense, an abstract description – as many of its components are replaced over time without affecting its identity as an individual organism. However, we would hardly say that "bacterium" is either a platonic abstract object or a mere descriptive abstraction, as this would imply that even you and I are abstract objects. Consequently, there is a sense in which abstraction is constitutive of what an individual is, given its individuation conditions. For this reason, I will speak of an additional constitutive criterion rather than a descriptive one.

leads from a detailed implementation explanation to the kind of abstract general causal structure we find in computational mechanistic explanations, Kuokkanen is right that this is the abstraction that matters to the MAC.

Let us dig deeper into the example of a neural population code. Suppose we are interested in the coding strategies employed by a neural population to track a distal stimulus. Suppose also that we have a multilevel explanation that includes details from the action potential of every neuron involved, such as the dynamics of their ion channels, and the neurons' patterns of connections, plus details about every synapse including the type of neurotransmitters involved, plus details about where our neural population is in a perceptual system, plus all the details about how stimuli are processed through the system's hierarchy. This multilevel hierarchy explains phenomena way beyond the coding strategies that were our original concern. So, for epistemic reasons, it is convenient to fix our views on the population level, how the neurons causally interact to activate each other, and maybe the evolution of the strength of the connection between them as a result of Hebbian learning. By doing so, we dispense with every detail about superior and inferior levels of the mechanistic hierarchy, thus performing vertical abstraction.

Furthermore, we still have some details that are not entirely useful to our inquiry about the coding strategy. We have our view fixed on two levels now, the level of the whole individual neurons and the level of the population. However, the specific voltage level in every neuron might not be important to us. We can represent them simply as mathematical Dirac deltas. We can even abstract away from the fact that they are neurons and focus specifically on their patterns of connections and activation. When we do so, we abstract within the same mechanistic level of organization, obtaining a computational explanation by performing horizontal abstraction.

Kuokkanen follows an important distinction emphasized by Elber-Dorozko and Shagrir (2021) between the computational hierarchy and the implementation hierarchy (for an additional example, see figure 1 in section 4). The computational hierarchy contains only mathematical descriptions of operations performed by putative sub-components of a system and their topology. The implementation hierarchy contains specific details about the biological traits or technological features of the components that perform those computations.

We can see that, following Kuokkanen's analysis, to a first approximation, a computational hierarchy is a horizontal descriptive abstraction of an implementation hierarchy. In our example, the implementation hierarchy is composed of the neurons as formed by particular biomaterials that have specific biochemical and biophysical properties omitted in horizontal abstraction. This is how Kuokkanen relates both explanatory hierarchies: he says there is only one hierarchy in the world. So, the implementation and the computational hierarchies aim to explain the same system in the world by employing different degrees of abstraction. As I will explain in more detail at the end of section 5, this formulation has some ambiguity. The reason is that, even when Kuokkanen addresses the generality of computational hierarchies, he claims that there is only one hierarchy that is both implementational and computational, depending on the degrees of descriptive abstraction deployed. However, the same computational hierarchy can usually be realized by more than one implementation hierarchy because computation is medium-independent. In this regard, Kuokkanen (2022a) argues that MAC is committed to what Elber-Dorozko and Shagrir (2019) call the single hierarchy view, as opposed to a separate hierarchies view that would imply that the computational and implementation hierarchies are separated and linked by a bridging relationship[6]. Instead, I will argue that the opposition between the separated

[6] This is how I interpret his article based on the following: (1) he refers to the account in these terms: "the arguments from explanatory value and multiple realizability are not real problems for the MAC as a single hierarchy view" (p. 370), and (2) the fourth section is entitled: "The non-problems and the problem of MAC as a single hierarchy view".

hierarchy view and the single hierarchy view is a false dilemma and that they are not wholly the same nor completely separated as to require a bridging relationship.

4. Higher-order mechanisms

As said, medium independence is crucial to MAC. In this section, I argue that medium-independent mechanisms have a different set of individuation and constitutive relevance conditions than implementation mechanisms[7]. Therefore, they should be conceptualized as supporting a different kind of mechanistic explanation, which I call *higher-order* explanation (section 4.2). The notion of order to be employed here is a descendent of Kim's (1998). Kim's orders were first opposed to mechanistic levels by Craver (2007, p. 165). Consequently, I will analyze the relationship between orders and Marr's levels of analysis and conclude that they can be integrated into the MAC as orders as opposed to levels. In the following section, I will use this notion of order to distinguish between a medium-independent functional property and a medium-independent mechanism.

I will briefly introduce Kim's view on orders in subsection 4.1 and propose an enriched version of this notion within MAC in order to accommodate computational explanation properly within a mechanistic framework.

4.1 Kim's orders and their relation to mechanisms

Kim (p. 20) introduces the notion of second-order property by means of the following definition:

[7] The notion of "implementation mechanism" is to be interpreted in the following manner: (1) they are the explanandum of a mechanistic explanation that includes more than mathematical and topological structure details, and (2) in addition, the target system of that explanation should be susceptible of a computational (medium-independent) mechanistic explanation – i.e., this target system's behavior has at least two possible mechanistic explanations – corresponding to the putative computation it implements.

F is a *second-order property* over set B of base (or first-order) properties iff F is the property of having some property P in B such that D(P), where D specifies a condition on members of B.

This definition can easily be extended to properties of any order, including higher than the second order, in the following way. If we substitute the set of properties B with a set of properties of order n, then we obtain the definition of an (n+1)-order property.

The example that Kim provides is that of having a primary color. Having a primary color is a condition D which applies over first-order color properties. Kim correctly claims that, normally, functional properties are higher-order properties in this sense. He also notoriously claims that higher-order properties are not causally efficacious but epiphenomenal, meaning that all the causal work is done by the first-order properties and, as a consequence, higher-order properties are redundant. This is known as the causal exclusion argument. Additionally, since Kim was not a mechanist, he did not have the conceptual tools to distinguish mechanistic levels from orders.

Since, as Kuokannen claims, levels and orders are linked to different sorts of mechanistic abstraction, namely, vertical and horizontal, the causal exclusion argument for higher levels should be responded to differently than the one for higher orders. I will not say anything about the level case, for I take Piccinini (2020) to say enough about it[8], and the main target of the present writing is the concept of order. Regarding orders, in section 4.2, I will emphasize that higher-order mechanisms should be formulated in terms of generalized basic components (GBCs), the way they are connected to each other – i.e., their topology – and the mathematical operations they perform.

Although higher-order properties could be seen as disjunctions of first-order properties – e.g., having primary color is equivalent to being red or being yellow or being

[8] Piccinini (2020, pp. 36-37) advances an egalitarian ontology of levels to deal with causal exclusion. According to him, higher-level properties are aspects of their realizer properties that are invariant under some changes.

blue – not every disjunction is equally valid when formulating mechanistic orders, those formulated in the way described above – using GBCs and their patterns of activities and connections – have relevance since they account for minimal causal patterns that allow for explanatory unity across a larger range. Notably, Craver (ibid.) introduces this distinction between mechanistic levels and orders of properties, which he equates to Marr's (1982/2010) levels of analysis. The main difference, he claims, is that in orders of properties what is realized and what does the realizing are not related by a part-whole relationship. In the example above, being red – a first-order property – and being of a primary color – a second-order one – are properties that can be assigned to the same red truck toy. In contrast, as we already saw, mechanistic levels are related as parts to wholes. Indeed, Craver is right that Marr's computational, algorithmic, and implementation levels of realization could all be applied to the same computational system as a whole.

Nevertheless, Craver says that he will exclude orders from the mechanistic taxonomy because any attempt to locate them within "would be contentious" (p. 165). In the next subsection, I will argue both that they can be located within the mechanistic taxonomy and that their location is relevant to the MAC.

4.2 Orders of properties, orders of mechanisms, and the MAC

I said above that orders of properties are relevant because medium-independent mechanisms have different individuation and constitutive relevance conditions than implementing mechanisms. It is convenient to analyze these conditions separately.

Let us start with individuation by considering the following example. Consider the case of a digital half-adder. We can explain its capacity by just appealing to its decomposition into logic gates, their capacities, and their interconnections. This is a computational explanation (C) since it does not appeal to any implementation property. Consider, in turn, a diagram of an electronic integrated circuit that implements a digital half-adder. It specifies the type of transistors and includes how they are connected and their source of electrical power. This electronic implementation (E) implements (C).

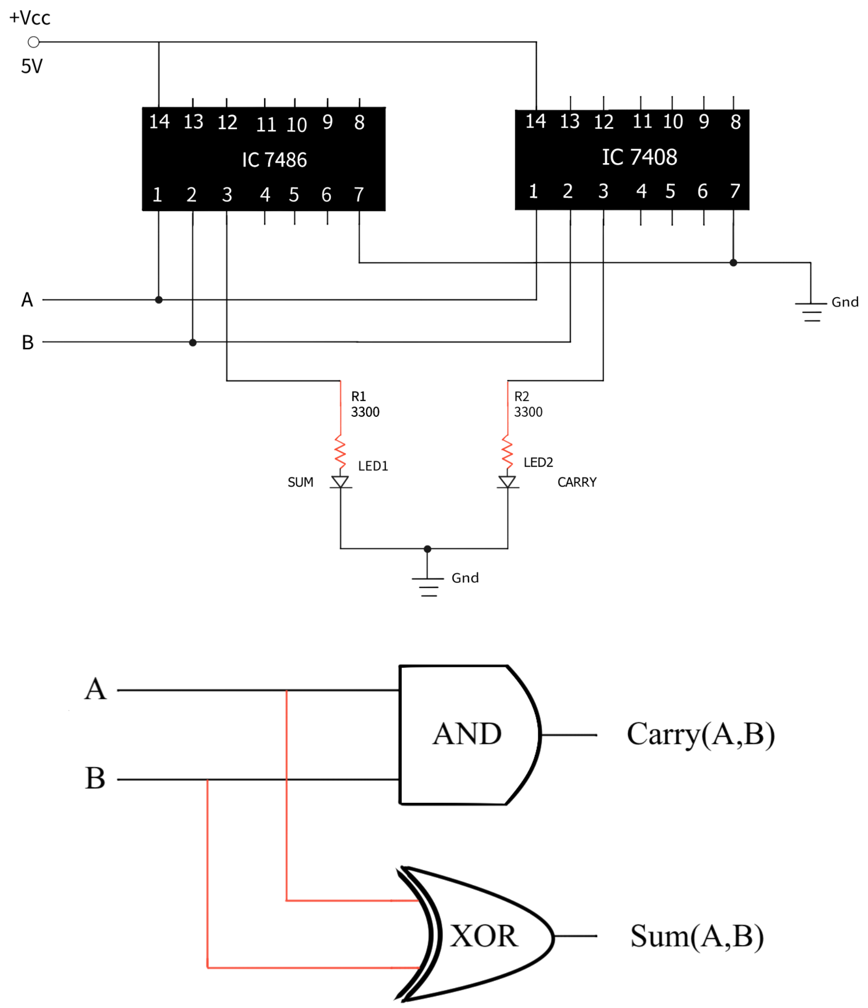

Figure 1: Below, the computational explanation (C) of a half-adder. On top is a possible implementation of it using integrated circuits (E); the outputs are delivered in terms of lights that can be on or off.

Furthermore, let us consider a diagram of a vacuum tube circuit that also implements a digital half-adder. It specifies the type of vacuum tubes that are to be connected to implement the computation. This vacuum tube circuit (V) also implements (C).

There is an ambiguity in Kuokkanen's theory of a single integrated hierarchy, namely, his view that the implementation hierarchy and the computational hierarchy are one and the same thing. If we are only provided with (C), can we decrease the level of horizontal abstraction to single out an implementation hierarchy? We cannot. If we try, we obtain a disjunction of possible implementation systems, (E), (V), or others. However, this does not happen in the other direction. We can easily determine the corresponding computational hierarchy (C) if we are provided only with (E) or (V). This generates an asymmetry between computation and implementation.

Consequently, the computational hierarchy cannot be characterized as linked to only one implementation hierarchy. There is no one-to-one relationship. Every computational hierarchy corresponds to a possibly vast disjunction of implementation hierarchies.

Let us consider a second example. The cellular automaton known as Conway's Game of Life is capable of implementing itself when considering a more extensive grid of metapixels (called OTCA metapixels after Outer Totalistic Cellular Automata Meta-Pixel) formed by 2,048 x 2,048 pixels each (Todesco, 2013). As a result, we obtain the same automaton, but now it is a version "2,048 times larger and 35,328 slower of the original one" (p. 241). The reason is that every metapixel gets actualized every 35,328 steps of the original one. Since this process is iterative, we can keep doing this indefinitely and obtain a potentially infinite hierarchy of computationally equivalent levels. Here every level is determined by the relation between the pixels and the metapixels in the next iteration.

What this second example shows is that, in some cases, in addition to the fact that computation cannot be identified with a specific implementation, it cannot be identified even with a specific level of a mechanistic hierarchy since it is multiply realized throughout it. So we have a potentially infinite disjunction of implementations of the same computational structure (G) across the same mechanistic hierarchy.

Is each of these disjunctions identifiable with a single mechanism? Chirimuuta (2014) claims that the mechanist must qualify them as mere sketches of mechanisms. By contrast, some mechanist philosophers insist that computational structures, such as C and G, have all the relevant causal details necessary to account for the computation (e.g., Fuentes, 2023; Piccinini, 2020; Wajnerman Paz, 2017b).

Piccinini (2020) addresses this concern in terms of two notions. Firstly, he maintains that (V), (E), and any other implementation of the half-adder share an aspect, so there is something invariant when considering the differences in the structures. Secondly, he defines a medium-independent mechanism as a multiply realizable mechanism whose vehicles are, in turn, multiply realizable. The question naturally arises about the nature of the multiply realizable mechanism. Is it one mechanism that could be realized in many different ways? Are they different mechanisms that share a standard structure that makes them the same in some sense? If so, what is this sense? To reinforce the point, (V) and (E) are different mechanistic kinds, but according to the idea of a multiply realizable mechanism, they are in some sense the same. Consequently, something else about individuation needs to be said to address skeptics about the viabiliy of mechanistic criteria for the individuation of computation (SIs). These remarks about individuation are not exclusive to computation or medium-independent mechanisms. It is easy to see that the same considerations could, in principle, apply to mousetraps.

Let us now consider what happens with constitutive relevance. The usual criteria for constitutive relevance in mechanistic explanation do not exhaust the epistemic norms for medium-independent mechanistic structures. Constitutive relevance criteria in the case of implementing mechanisms have been successfully accounted for by Craver et al. (2021). The criterion they arrive at is called matched interlevel experiments. It is the following:

> **(MIE)** To establish that an entity X and its activity ϕ are constitutively relevant to a mechanism that ψs, the following experimental results and matching condition are jointly sufficient:
>
> (CR1i) If an experiment initiates conditions ψin while a bottom-up intervention, I, prevents or inhibits X's ϕ-ing, alterations to or prevention of ψ's terminal conditions, ψout, are detected.
>
> (CR1e) If a bottom-up intervention, I, stimulates X's ϕ-ing, ψ's terminal conditions, ψout are detected.
>
> (CR2*) If a top-down experiment initiates conditions ψin and detects ψ's terminal conditions ψout, X's ϕ-ing is also detected.
>
> (Matching) The activities ϕ activated or inhibited in bottom-up experiments (CR1i and CR1e) must be of the same kind as, and occur within quantitatively overlapping ranges with, the activities ϕ detected in top-down experiments (CR2*).

Now, MIE establishes whether an entity is part of a mechanism for a phenomenon or function. As a criterion, MIE works well as an epistemic rule for implementation mechanisms, but it is not the same with medium-independent ones. The reason is that every experimental intervention must be performed on a specific realization or implementation of the structure. Thus, if we apply MIE while experimenting with some kinds of logic gates (electromechanical, electronic, made of vacuum tubes, etc.), it will tell us that a medium-dependent component is part of the mechanism, while if we apply it to others, different component entities will emerge. In other words, interventions on (and interventions that affect) properties that belong to the medium will always pass the MIE test and, therefore, would be incorporated into the phenomenon's model.

By contrast, in computational mechanistic explanations, we need an additional criterion which aims not to identify maximally concrete parts and activities of the mechanism but rather to determine which details should be left behind when abstracting the medium-independent aspects of it. This criterion must be mainly an omission criterion, i.e., one used for identifying the details to be omitted in higher-order explanations. Furthermore, this criterion must accord with the desiderata of both describing a relevant mechanism in terms of common abstract components and activities and, at the same time, making explanations in computational sciences as general as possible. Let's dig deeper into this second desideratum and its relation to the philosophy of explanation.

Kitcher (1981) famously proposed that some scientific explanations could be unified to explain a higher number of phenomena resorting to a small set of patterns of arguments. The main epistemic value behind this approach is to minimize the number of explanantia while maximizing the number of explananda. However, Kitcher's conception of scientific unification is tied to the covering-law model of explanation. In this model, scientific explanation is perceived as an argument derived from general laws and initial conditions. As has been suggested throughout the development of the new mechanistic philosophy (e.g., Craver, 2007), the epistemic norms of the covering-law model are not consistent with the scientific practices in many special sciences. More specifically, they are not consistent with cognitive neuroscience's explanations and also, at least according to the MAC, with explanations in the computational sciences.

For these reasons, Levy (2016) adapts Kitcher's unificationist model to cognitive neuroscience. In doing so, he identifies three main species of explanatory patterns, network motifs (Levy & Bechtel, 2013), design principles (Sterling & Laughlin, 2015), and canonical neural computations (CNCs) (Carandini & Heeger, 2013). I will focus on the latter two[9].

[9] The reason is that with respect to motifs, Levy (2016) offers only the following conditional claim: "if, and to the extent that motif-based analysis proves important and informative, it will be a distinctively non-level-like way of thinking about the brain".

Design principles are general principles followed by neural systems that optimize a given resource variable in a biological system for the sake of the organism's survival. Fuentes (2023) offers a way of integrating optimality explanations of this sort with neurocomputational mechanistic explanations. According to him, efficient teleofunctional mechanisms are mechanisms that not only fulfill a teleofunction but also do it through an optimal strategy, i.e., this strategy optimizes a tradeoff that maximizes performance variables while minimizing resources expended. Fuentes argues that CNCs are themselves efficient teleofunctional mechanisms, i.e., they are shaped by design principles. For this reason, we can focus our analysis on the flat unity provided by CNCs, as they describe efficient generalized basic components for neural computation.

In the same spirit that motivates Levy, Wajnerman Paz (2017a) proposes an epistemic value that he calls *horizontal integration*. Wajnerman Paz speaks of horizontal integration as a kind of explanatory integration that aims to target multiple phenomena by combining a small basic set of canonic components in different ways. These basic components, in the case of neural computation, would be CNCs.

Can we find something akin to canonical neural computation in other kinds of computation? According to the MAC, we know of at least three different species of computation, namely, neural, digital, and analog (Piccinini & Bahar, 2013). Although they are not supposed to exhaust the space of possible kinds of computation, an analysis of what happens in each case will give an idea of how mechanistic computational explanations work. On the one hand, digital computation can be decomposed by using logic gates as basic components that process digital vehicles and their patterns of connections – it is important to remember that this is only a specific formalism to decompose digital computations; cellular automata, which decompose into cells and their rules of interaction, are also a form of digital computation. On the other hand, analog computation can be decomposed in terms of integrators, as the main basic components that process analog vehicles, other components, and their arrangement.

In light of the above, I propose the following generalized basic components (GBC) criterion as an epistemic norm based on Wajnerman Paz's horizontal integration epistemic value:

> **GBC:** A mechanistic computational explanation of a system M, with M being one of a class of computational mechanisms K defined by a decompositional formalism F, should be formulated or formulable in terms of a set of generalized basic components G(K,F) that are characteristic of the computational explanations of the mechanisms in K when described under formalism F.

Here, K could be neural networks, digital computers, analog computers, or another class. G(K,F) is the set of components in which explanations that appeal to K are formulated. G(K,F) also depends on decompositional formalism F, as digital computational systems can be decomposed into logic gates arrangements or cellular automata grids, where G(K,F) is formed by logic gates or cellular automata cells and their rules of interaction, etc. Moreover, the formalism employed, and therefore G(K,F), is different in different orders of mechanisms, as the examples below will make clear. For instance, in digital computation, G consists of different logic gates. Moreover, given that G(K,F) is a set of basic computational components, its members must satisfy the mechanistic definition of computation. In other words, they must be medium-independent and have the teleofunction of manipulating multiply realizable vehicles according to a rule (Piccinini, 2020).

Why do I claim that GBC acts as an epistemic norm? Firstly, as I stated above, it emanates from an epistemic value, namely horizontal integration. Secondly, and perhaps more importantly, it governs the kind of components that mechanistic computational explanations can make use of. They should be generalized across a class of computational mechanisms under a specific formalism –e.g., digital computation explanations under the logic gates formalism should be formulated in terms of logic gates and their pattern of connections, neural networks computational explanations should be formulated using neurons[10] and their weighted connections, etc. Moreover, this epistemic norm has ontological relevance, as it states the kind of generalized entity that is valid to pose when performing computational explanations.

As an additional virtue of the formulation, this norm also allows us to elucidate the line between physical computation and purely mathematical computation. Physical computation has to do with generalized basic components intended to unify a large number of maximally concrete mechanistic explanations. Components that are, in principle, non-susceptible of physical implementation, such as unbounded tapes (see section 5), should not be considered part of a physical theory of computation.

I need to say a bit more about how GBC is an additional constitutive relevance clause not implicit in the MIE criterion. Firstly, GBC presupposes MIE. Once we have the components and properties of a variety of mechanisms, GBC is concerned with specifying a *set* of generalized basic components to unify explanations. Thus, to find generalized basic components, we need mechanistic first-order explanations which possess these common aspects. Second, the epistemic role that MIE still plays in computational explanation is another reason to deem it mechanistic. On the one hand, computational explanations are mechanistic because they are structured in levels formed by components and their activities and interactions in a minimally detailed causal structure. On the other hand, they are mechanistic because they depend upon the same epistemological norm common to mechanistic explanations, namely, MIE.

[10] Here 'neuron' is not characterized in terms of specific biophysical properties but in a medium medium-independent way.

Furthermore, since, as I argued, medium-independent mechanisms have different conditions of individuation and constitutive relevance, they both deserve to be classified as an essentially different class of mechanisms than implementation ones. To be sure, both kinds are mechanisms in the sense of being constituted by their parts and their activities and the relationships between them. Nevertheless, their different individuation and epistemic status require subclassification. I will call medium-independent abstract structures *higher-order mechanisms* and implementation mechanisms first-*order mechanisms*.

Before we move on, I must say something more about the notion of a mechanism being higher order. Up to this point, I have argued for the existence of at least two orders based solely on differences between the constitutive and individuation conditions they are subject to. One question remains. Are there more than two orders? My response is that it depends on the system under consideration. In some cases, when there are more than two, it is controversial to distinguish which of them are purely computational and which are purely implementational. For this reason, I will provide three examples of what these orders are.

A first approximation to the notion of orders I am articulating here is to consider them, as Craver (2007) suggested, in connection with Marr's (1982/2010) three levels of analysis. Marr's three level of analysis provides us with at least three different orders, the computational, the algorithmic, and the implementation, which in many cases are all we need to account for a computational system. However, a revision of their names is needed, as will be evidenced below.

The first example is the half-adder described above. E is a first-order mechanism that implements half-addition in an electronic circuit. By contrast, C is a second-order abstract mechanism. Its parts are logic gates. Although we can say that this corresponds to Marr's algorithmic level, this is somewhat misleading. First of all, the concept of algorithm does not presuppose that the different sub-operations are carried by parts, which is what C describes.

Also, the concept of an algorithm gives the gist of digital computations[11], whereas orders are more general, as will be elucidated in the next example. As C describes the tasks the computational strategy assigns to each part and how they are related to compute the function in a medium-independent way, this is a computational order. I will call it the *computational strategy* order. Here G(K,F) members are logic gates. The third order is the most horizontally abstract. This order only describes the mathematical function that rules the input-output transition. This is how Piccinini (2020) characterizes the exponentiator described above (see section 2). In the case of the half adder, it would be the function $H:(x,y)\rightarrow(z,t)$ where z represents the last digit of the sum, in binary notation, of x and y, and t represents the carry. This is what Marr called the computational level of analysis, but since the second-order level is also computational, as we saw, we will call it the *mathematical function* order. This order is the least restrictive in terms of individuation. All that it requires is that there be two degrees of freedom with two distinguishable states that can be classified as inputs, any physical structure that allows the relevant mathematical transition, and two degrees of freedom with two distinguishable states that act as outputs. This order possesses the highest degree of generality as it does not require, as a condition of individuation, applying the strategy in the second order as long as the mathematical function displayed in Table 1 is computed. It is important to discuss this order in more detail with respect to horizontal integration and the norms of mechanistic explanation.

[11] Algorithms are usually defined as effective procedures to compute the values of specific functions. For example, the algorithm employed to multiply two numbers that have more than one digit each, or, the algorithm used to obtain the least common multiple, which is usually taught in school.

x	y	z = sum(x,y)	t = carry(x,y)
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

Table 1: Half-addition input-output rule

The second example is a differential analyzer. In the first order, we have to describe one of its implementations. We can say that it is constituted by a specific arrangement of rigid discs connected thus and so. In the second order, we can describe every single integrator in an abstract way, as they are described by the corresponding G(K,F) set, and specify the arrangement they form to solve a (system of) differential equation(s). We also need to specify which degrees of freedom are vehicles for the inputs, i.e., the initial and boundary conditions, which degrees of freedome are vehicles for the output, and the function *f*, which is the solution of the equation on a given interval. Finally, in the third order, the mathematical function order, we specify only the function E: $(e, b) \rightarrow f$. This gives us the solution *f* of the differential equation *e* given the initial and boundary conditions *b*.

What happens with G(K,F) in the mathematical function order? Is this order truly mechanistic in the sense of the MAC? In order to address these questions, we need to introduce a distinction – which is novel and missing from the literature – between a medium-independent functional property and a medium-independent mechanism. Properly speaking, an unqualified mousetrap is not a multiply realizable mechanism, and Piccinini's exponentiator is not a medium-independent mechanism. They are, in turn, multiply realizable and medium-independent functional properties as no mechanistic decomposition in component parts and activities is offered. This medium-independent functional property corresponds to a phenomenon explanandum that needs to be mechanistically explained. However, it is not itself a mechanistic explanation because neither implementational parts nor medium-independent GBCs are offered. Since this medium-independent functional property is obtained by exerting the highest level of horizontal abstraction, as Kukkanen notes, I will call it the highest order by analogy[12]. The difference with the numbered orders (first order, second order, etc.) is that the highest order does not offer a mechanism, as it is not subject to GBC, so it is not mechanistically explanatory. However, its attribution to a physical computing system can be empirically adequate or inadequate to the extent that the system is actually computing the purported medium-independent function.

This is not to say that finding adequate phenomena for mechanisms to explain them is a trivial task, as some have suggested that mechanism implies (e.g., Shagrir & Bechtel, 2017; Carrillo & Knuuttila, 2023), but that in order to have an explanation, we still need to know how the target system decomposes the computation and different tasks are assigned to different parts of it.

To sum up, higher-order mechanisms are explanatory and formulated in terms of GBCs. The corresponding mechanisms are called second-order mechanisms, third-order mechanisms, and so on. By contrast, medium-independent functional properties can be identified by analogy with the highest order, as no further horizontal abstraction over a mechanism is possible. However, they are not mechanistically explanatory[13], even when their adjudication to a physical system can be assessed as empirically adequate or inadequate. Piccinini's exponentiator is a medium-independent functional property in this sense. By contrast, hierarchy C, in the half-adder example above, is a genuine medium-independent mechanism.

[12] Higher orders are obtained by horizontal abstraction. However, what I call the highest order is technically not an order properly since it is not formulated in terms of a mechanism using GBCs. Nonetheless, I have decided to call it that way by analogy with the orders since it corresponds to the maximum possible level of horizontal abstraction.

[13] Although they can be part of mechanistic explanations, as explanandum phenomena.

Thus, GBC synthesizes the epistemic rules that, in addition to MIE, are necessary for every mechanistic explanation with an order that is higher than the first.

In the two examples above, is the second order computational or implementational? It certainly says something about how the function has to be implemented if it falls under its mechanistic description. As said above, this inspired Marr to only call computational the third order, namely, the one that describes the medium-independent functional property but not the mechanistic decomposition. But, as this second order depicts a medium-independent mechanistic structure that does not contain any detail about the specific material implementation, we are justified to also call it computational.

Moreover, some computational systems could eventually be explained in more than three orders of abstraction. A common digital computer could be described in the order of its medium-dependent technology, the assembly language order, the operating system order, etcetera, until reaching the last order describing only the dependence between inputs and outputs.

5. Computing mechanisms and their medium independence

Computational mechanistic explanations decompose medium-independent teleofunctions into more basic ones carried out by GBCs. The difference, between a medium-independent functional property and a medium-independent mechanism is that the first only describes the explanandum phenomenon of computational mechanistic explanations, while the latter offers a mechanism with its parts, activities, and relationships. Nevertheless, an important question remains: in virtue of what are these functional properties and GBCs medium-independent and, more importantly, computational? Although GBCs cannot be further computationally decomposed in a computational mechanistic explanation, since they are basic, they need to have a minimal internal structure if they are to be considered computational. Let us inquire into what that structure amounts to.

The problem here is to ensure that GBCs are, in fact, computational and not merely multiply realizable. Since they are basic, we cannot appeal to more basic computational structures from which they are composed. We can address this issue by appealing to Piccinini's (2015, 2020) definition of a teleofunctional computational property (see section 2), i.e., multiply realizable mechanisms in which both the inputs and the outputs are multiply realizable and have the teleofunction of manipulating them according to a rule.

Now, this amounts to a minimal structure in which there is something that can be treated as an input, another physical system or property that can act as an output, and a mechanism that guides the transition. In some mechanisms, the input and the output are different properties of the same physical system. For example, in a Turing Machine, both the inputs and the outputs are inscriptions on portions of the tape, or in mousetraps, the free mouse and the trapped mouse are properties of the same mouse. However, in other cases, the inputs and outputs are of different sorts. For example, in music boxes, different patterns inscribed in metal cylinders, the inputs, give rise to different melodies as outputs. But, as we know, only the first of these three examples counts as a medium-independent mechanism. The reason is that only the Turing Machine can act over digits that are multiply realizable without losing its functional identity as a Turing Machine. Mousetraps are constrained to work over mice and music boxes to produce sounds. Finally, some mechanisms are not input-output in any relevant sense.

Therefore, a medium-independent property is obtained when:

(1) a special kind of mechanism, namely, an input-output mechanism, is horizontally abstracted so as to obtain a higher-order mechanism, a functional rule, or a GBC.

(2) Its inputs and outputs are also horizontally abstracted so as to be multiply realizable.

(3) (**Teleofunctional retention**) The teleofunctional identity is retained[14].

If we have a mechanistic decomposition in terms of parts and activities of the input-output mechanism described in (1) in terms of GBCs and their activities, we have not only a medium-independent property but a medium-independent mechanism.

We also need to know how to apply these specifications to a concrete system to decide if it is implementational. As when we ask ourselves, is the brain a computer? To ask if a particular mechanism is a computer is really to ask whether it implements computations in an observer-independent way. As said above, the implementation order of a mechanistic hierarchy is that which carries less horizontal abstraction and more detail. It includes specific biophysical or technological details about a computational system. It is the order where the usual conditions of individuation and constitutive relevance criteria for mechanisms, i.e., MIE, are sufficient.

When are we allowed to state that a first-order mechanism implements a computation? This happens when a horizontal abstraction of it is a medium-independent computing mechanism or a GBC. The circuit of a half-adder implements a computation because, when horizontally abstracted, we obtain C. C is computational because it is formulated in terms of GBCs, namely, logic gates. The teleofunction of logic gates is to manipulate vehicles according to a rule as the MAC requires. More strongly, C is a computational mechanism and does not merely describe a computational teleofunction as the highest order does (see section 4), as it decomposes the computation in terms of GBCs.

[14] In the above example, if we were to abstract away the fact that the music box produces sounds and preserve only the mathematical structure of those sounds, we would not **retain** its teleofunctional identity as a music box. More precisely, its identity as a mechanism whose teleological function is to produce sounds from input patterns would be lost.

Computational mechanisms are higher-order mechanistic structures. On the one hand, this means they are decontextualized from the lower-level hierarchy or hierarchies that gave them origin so that their conditions of individuation do not depend upon them. In other words, the ***same*** computational mechanisms could be implemented by many radically different first-order mechanisms. On the other hand, this means that what is epistemically relevant when assessing whether a component is part of the mechanism and whether the level of detail in the description is appropriate is that the mechanistic explanation must be capable of accounting for the teleological function of the system in a minimal sense. For example, in the digital half-adder, the level of detail must account for the teleofunction of half-adding in terms of the teleofunctions of the GBCs in the formalism under consideration, i.e., the logic gates.

Finally, how do the computational and the implementation hierarchies relate? As already stated at the end of section 3, Kuokkanen echoes a dilemma, proposed by Elber-Dorozko and Shagrir (2019), between a single hierarchy view and a separated hierarchies view of computational mechanisms. He opts for defending the single hierarchy view as he considers that the only distinction between implementational mechanisms and computational ones is one of degree of horizontal abstraction. One problem posed to a separated hierarchies view is that it would require a bridging relationship that cannot be considered mechanistic, since it is not part-whole but less abstract to more abstract (cf. Coelho Mollo, 2018; Elber-Dorozko & Shagrir, 2019; Shagrir, 2022). However, I take this dilemma to be a false one, since the hierarchies are neither completely separated nor completely the same.

For a computation to be physical, there must be at least one instance in which it is or could be implemented. In other words, the mechanistic computational structure needs an implementation structure from which it is horizontally abstracted. In this sense, the half-adder is a physical computing system, but the Universal Turing Machine with an unbounded tape is not (although bounded versions are). The latter is a purely mathematical computational device. Thus, even when the computational hierarchy is decontextualized, it is the computational hierarchy of at least one implementation.

We do not need a bridge law as the computational hierarchy is a horizontally abstracted description of one or many implementation systems. But it cannot be considered the same given that its explanatory force covers more systems in the world. As a consequence, they are not the same, but they are not completely separated as physical computational hierarchies depend on implementational ones.

6. Answering MAC's critics

In this section, I answer the criticisms posed by the four groups of MAC critics listed in section 2 using the distinctions articulated above. This shows the relevance of the proposal introduced in sections 4 and 5. Some remarks, concerning the non-essentiality of semantic properties to physical computation are provided at the end.

In response to skeptics about abstract mechanisms (SAs), we can say that horizontally abstract mechanistic explanations capture aspects of the world's causal structure. They track the specifics of how a mathematical computation is decomposed into sub-computations when performed by a physical system. For example, an analog computing mechanism will track how a specific arrangement of integrators is composed to compute the solution of a differential equation given certain initial and boundary conditions.

Furthermore, even in first-order mechanistic explanations, some minimal degree of horizontal abstraction is required, so the more details the better (MDB) principle, posed by Chirimuuta's mischaracterization of mechanism, cannot be met[15]. Hence, not only is MDB

[15] One can still argue that, while the requirement is impossible to meet, it could act as a regulative epistemic ideal, in the sense that it would be better to have no abstraction at all (thanks to an anonymous reviewer for making this point). However, this would conflict with the generality of the explanations in science, and, for that

not necessary for mechanistic explanation, but it also conflicts with its nature. For example, when we say that the hippocampus plays a central role in memory encoding, we do not refer to a specific subject's hippocampus. Moreover, in many mechanistic explanations in neuroscience, even the species or, more strongly, even the animal class (e.g., mammal) in which the experimental data was obtained is omitted. The reason is that neuroscientists assume that there is something common in the functioning of a trait across species and classes.

The MDB condition arises because of a misunderstanding of the difference between the notions of mechanism sketch and mechanistic explanation. When the description is put in black box terms, we need more detail about the decomposition of the capacity. But, once we have a valid decomposition, the main difference between sketch and explanation is whether that decomposition tracks *how* the capacity is *actually* decomposed in the target system – in sub-activities of subcomponents – or a *how possibly* proposal of the way the researchers believe –often based on behavioral data – that the capacity could be decomposed in the target (Piccinini & Craver, 2011). No condition about the level of detail is required, as is eloquently summarized by Kaplan (2011, p. 347) in his 3M condition:

> (3M) A model of a target phenomenon explains that phenomenon to the extent that (a) the variables in the model correspond to identifiable components, activities, and organizational features of the target mechanism that produces, maintains, or underlies the phenomenon, and (b) the (perhaps mathematical) dependencies posited among these (perhaps mathematical) variables in the model correspond to causal relations among the components of the target mechanism.

Once we have a how actually explanation, the level of detail will be determined not by MDB but by other factors, such as horizontal integration (see section 4). As Wajnerman

reason, would be irrational for the mechanist to maintain such an epistemic norm. In the most extreme case, horizontal integration and the GBC norm for higher-order mechanisms, we go exactly in the opposite direction.

Paz (2017a; 2017b) argues, in what I have called higher-order mechanisms, details are left out not because of ignorance but because of the epistemic value of horizontal integration.

In response to skeptics about medium-independent teleological functions (SFs), it could be argued that higher-order mechanisms can be functional in the sense required by the MAC. Consider a Turing Machine with a bounded tape of sufficient length. Each particular instance of a Turing machine could contribute to different goals or none if not put into practical use. However, the goal they contribute to in specific instances is not the function. For example, when we say that a certain software helps an engineer to design durable structures, designing durable structures is the goal, but calculating physical parameters is the function of the software. By performing digital operations, Turing machines could contribute to a variety of different goals. In all of these cases, the contributions are made by means of the same causal role F, which is to perform those digital operations. Hence, performing those digital operations is its function. If that is so, teleofunctionality is not restricted to implementing mechanistic structures, as medium-independent mechanisms could be teleofunctional as well.

When addressing skeptics about medium-independent vehicles (SVs), it is important to bear in mind that there is an evolution in the MAC from (Piccinini, 2015) to (Piccinini, 2020). This is of the most importance since, in the recent formulation, the concept of a medium-independent vehicle, which SVs consider flawed, has been further enriched. In the new formulation, we have seen that it is important to MAC that the mechanism and the vehicles it processes be multiply realizable. Therefore, "medium-independent vehicle" means multiply realizable vehicle manipulated by a multiply realizable mechanism. Every mention of specific dimensions of variations has been erased and replaced with the notion of a multiply realizable vehicle[16]. Maley (forthcoming) claims that, if the criterion for a vehicle to

[16] The reason is not that "dimensions of variation" is intrinsically wrong, but that it could lead to the kind of misunderstanding accounted for here. If, by contrast, dimensions of variations are interpreted as pertaining to degrees of freedom (such as digits) – digits are individuated only by the capacity of the physical system to

be medium-independent is responsiveness only to specific dimensions of variations, then a cylinder lock will count as computational. The reason is that the cylinder lock is sensitive only to the geometric pattern on a key so it can be said to calculate the characteristic function *K* of a specific pattern *p*. This is the function that delivers 1 (open) when a specific pattern is applied via the corresponding key and 0 (closed) in every other case.

The problem here, as I said above, is that, to be a medium-independent vehicle, the new definition requires not only sensitivity in specific physical dimensions, in this case, shape, but also that the mechanism and vehicles be multiply realizable. Cylinder locks, as such, are not a multiply realizable mechanism since this would require that different implementations of a cylinder lock use a relevant property other than shape. Relevant properties are properties that matter for the functioning of a mechanism. For example, electrical current is a relevant property in an integrated circuit, but color is not. Relevant properties, says Piccinini (2020), distinguish mere variable realizability from genuine multiple realizability (Piccinini, 2020). But if it were to process other relevant properties, it would cease to be a cylinder lock. Therefore, the cylinder lock is not a higher-order computational mechanism. Can it be used to implement computations? This is to ask whether a horizontal abstraction of it can count as a higher-order computational mechanism.

The answer depends on the width of the teleofunction under consideration. Teleofunctions can be construed in various degrees of width, from the narrowest to the broadest (see Anderson and Piccinini, forthcoming). For example, in the narrowest sense, the function of the heart is to expand and contract in a specific way, but in the broadest sense, the function of the heart is to transport nutrients and oxygen to every cell in the body in which it forms part.

distinguish a finite number of states from each other, and no reference to a physical variable is needed – rather than specific physical variables (such as shape), the problem vanishes (Piccinini, personal communication).

I believe that Maley's cylinder lock example illustrates something important about the nature of computational functions. As we saw in the previous section, we are allowed to say that a given artifact or biological trait implements a computation when a horizontal abstraction of it gives rise to a higher-order mechanism or GBC corresponding to that computation. If that is not possible, then we are not in the presence of an implementing lower-order mechanism.

In a narrow sense, the cylinder lock's function could be conceived of as rotating when all the cotter pins are in place. Thus, every cotter pin is used to identify whether a specific part of the key fits the geometric pattern required. This decomposition could be seen as decomposing the computation of the characteristic of the pattern into simpler computations that identify a specific part of the key. However, given that the cylinder lock has the capacity to implement the computation of the mathematical function *K*, the important thing here is whether it has the teleofunction to do so. Strikingly, the answer appears to be yes under some narrower functional width. For the contribution to an organism's goal of maintaining a door closed to strangers of the cylinder lock is precisely achieved by means of the causal role of producing the output "open" if and only if the right pattern is presented and by means of this operation letting only the carriers of the right key to access the protected content. That is, the goal is reached by means of the causal role of applying the characteristic function K. Another realization of this mechanism would be a magnetic card reader attached to a lock that responds to the corresponding pattern. Since both the mechanism (cylinder lock, magnetic card reader, etcetera), the inputs (highness, magnetic pattern, etcetera), and the outputs (a deadbolt stays or is displaced, an electromagnet lock stays on or is turned off, some data in a computer is unlocked, etcetera), we are in the presence of a genuine medium-independent computational mechanism.

However, Maley has gone too far in maintaining that this would be the ruin of the MAC. To see why that is so, let us consider another counterexample he proposes against MAC, his impact rotor sprinkler. Maley correctly points out that, in a rotor sprinkler, the angular velocity depends solely on the specific dimension of water pressure. Let us call *A* the function that delivers the right angular velocity given a pressure *p* on the sprinkler rotor *R*. Can the sprinkler rotor mechanism be considered a computational implementation of the mathematical function *A*? The answer is, clearly no. And, contrary to what Maley implies, the MAC has a straightforward explanation. The critical difference with the previous example is that the teleofunctionality of the rotor sprinkler does not depend in any relevant way on the angular velocity within a wide range of possibilities.

A much more radical example is posed by Polger & Shapiro (forthcoming). They argue that, since medium independence depends on what I have called above the width of the description of the teleofunction, namely, the way in which we describe the system and its relation to its environment, we could consider that the water of the ocean is medium-independent provided that we describe it as *whatever fills the oceans*. While water is essential to the survival of organisms in the oceans, filling the ocean is not a teleofunction in any relevant sense. Furthermore, water or any liquid filling the ocean is clearly not decomposable in any sense different from the way an aggregate is, contradicting Craver's (2007) condition for mechanisms. Finally, *whatever fills the ocean* is not a property ascribed to input-output systems. Therefore, the putative medium-independent property does not satisfy any of the criteria posed in section 5.

The moral here is that the notions of teleofunctionality and medium independence are inseparably bound together in the MAC. This explains why the cylinder lock can be considered implementational under specific functional widths while the rotor sprinkler cannot.

Finally, in response to skeptics about the viabiliy of mechanistic criteria for the individuation of computation (SIs), lower-order mechanistic individuation is not the kind of individuation relevant to computational neuroscience (e.g., Priorelli & Stoianov, 2023) and computer science (as in the half-adder above). In contrast, higher-order mechanistic individuation is the kind of individuation that we find in those sciences since, as we saw in higher-order mechanistic individuation, two implementations of the same abstract computational structure count as the ***same***.

Both SVs and SIs formulate their skeptical remarks as a means to argue for a semantic account of computation. According to semantic accounts, semantic properties are essential to physical computing systems. MAC claims that, although semantic properties are sufficient for computation, they are not necessary. If the considerations portrayed throughout the current section are correct, then, at least concerning its entailment from the skeptical arguments SVs and SIs, semantic properties are esentially involved neither in the definition of a physical computing system nor in the individuation of computations.

7. Conclusion

I have addressed some criticisms about the viability of the MAC. I have argued that when we dig deeper into MAC's foundational concepts, we can answer each one of them.

With that aim, we saw that mechanistic abstraction could be directed in two different directions, namely, vertical and horizontal, and that the most relevant for computation is horizontal abstraction. Moreover, since abstract mechanistic hierarchies are guided by different criteria of individuation and constitutive relevance, I named medium-independent mechanisms *higher-order* mechanisms. Higher-order mechanisms are key to computational neuroscience as they provide horizontal integration.

Once that was done, it was possible to appreciate properly the nature of computational mechanistic explanation, how it relates to implementation explanations, the nature of abstract mechanism's teleofunctions, and what is sameness in computational mechanisms.

REFERENCES

Anderson, N. G., and Piccinini, G. (forthcoming). *The Physical Signature of Computation: A Robust Mapping Account*. Oxford University Press.

Bechtel, W. (2009). Looking down, around, and up: Mechanistic explanation in psychology. *Philosophical Psychology*, *22*(5), 543–564. https://doi.org/10.1080/09515080903238948

Boone, W., & Piccinini, G. (2016). Mechanistic Abstraction. *Philosophy of Science*, *83*(5), 686–697. https://doi.org/10.1086/687855

Carandini, M., & Heeger, D. J. (2012). Normalization as a canonical neural computation. *Nature Reviews Neuroscience*, *13*(1), 51–62. https://doi.org/10.1038/nrn3136

Carrillo, N., & Knuuttila, T. (2023). Mechanisms and the problem of abstract models. *European Journal for Philosophy of Science*, *13*(3), 27. https://doi.org/10.1007/s13194-023-00530-z

Chalmers, D. J. (1996). Does a rock implement every finite-state automaton? *Synthese*, *108*(3), 309–333. https://doi.org/10.1007/BF00413692

Chirimuuta, M. (2014). Minimal models and canonical neural computations: The distinctness of computational explanation in neuroscience. *Synthese*, *191*(2), 127–153. https://doi.org/10.1007/s11229-013-0369-y

Chirimuuta, M. (2018). Explanation in Computational Neuroscience: Causal and Non-causal. *The British Journal for the Philosophy of Science*, *69*(3), 849–880. https://doi.org/10.1093/bjps/axw034

Coelho Mollo, D. (2018). Functional individuation, mechanistic implementation: The proper way of seeing the mechanistic view of concrete computation. *Synthese*, *195*(8), 3477–3497. https://doi.org/10.1007/s11229-017-1380-5

Coelho Mollo, D. (2019). Are There Teleological Functions to Compute? *Philosophy of Science*, *86*(3), 431–452. https://doi.org/10.1086/703554

Craver, C. F. (2007). Explaining the Brain. Oxford University Press. https://doi.org/10.1093/acprof:oso/9780199299317.001.0001

Craver, C. F., Glennan, S., & Povich, M. (2021). Constitutive relevance & mutual manipulability revisited. *Synthese*, *199*(3–4), pp. 8807–8828. https://doi.org/10.1007/s11229-021-03183-8

Cummins, R. (1989). *Meaning and Mental Representation*. Cambridge, MA: MIT Press.

Elber-Dorozko, L., & Shagrir, O. (2021). Integrating computation into the mechanistic hierarchy in the cognitive and neural sciences. *Synthese*, *199*(S1), 43–66. https://doi.org/10.1007/s11229-019-02230-9

Fresco, N. (2010). Explaining Computation Without Semantics: Keeping it Simple. *Minds and Machines*, *20*(2), 165–181. https://doi.org/10.1007/s11023-010-9199-6

Fresco, N. (2014). Physical Computation and Cognitive Science (1st ed. 2014). Springer Berlin Heidelberg : Imprint: Springer. https://doi.org/10.1007/978-3-642-41375-9

Fuentes, J. I. (2023). Efficient Mechanisms. *Philosophical Psychology*. https://doi.org/10.1080/09515089.2023.2193216

Garson, J. (2013). The Functional Sense of Mechanism. *Philosophy of Science*, *80*, 317–333. https://doi.org/10.1086/671173

Glennan, S. (2017). *The new mechanical philosophy* (First edition). Oxford University Press.

Haimovici, S. (2013). A problem for the mechanistic account of computation. *Journal of Cognitive Science*, *14*(2), 151–181.

Kaplan, D. M. (2011). Explanation and description in computational neuroscience. *Synthese*, *183*(3), 339–373. https://doi.org/10.1007/s11229-011-9970-0

Kim, J. (1998). Mind in a physical world. Cambridge, MA: MIT Press.

Kitcher, P. (1981). Explanatory Unification. *Philosophy of Science*, *48*(4), 507–531. https://doi.org/10.1086/289019

Kuokkanen, J. (2022a). No computation without implementation? A potential problem for the single hierarchy view of physical computation. *Synthese*, *200*(5), 370. https://doi.org/10.1007/s11229-022-03696-w

Kuokkanen, J. (2022b). Vertical-horizontal distinction in resolving the abstraction, hierarchy, and generality problems of the mechanistic account of physical computation. *Synthese*, *200*(3), 247. https://doi.org/10.1007/s11229-022-03725-8

Levy, A. (2016). The unity of neuroscience: A flat view. *Synthese*, *193*(12), 3843–3863. https://doi.org/10.1007/s11229-016-1256-0

Levy, A., & Bechtel, W. (2013). Abstraction and the Organization of Mechanisms. *Philosophy of Science*, *80*(2), 241–261. https://doi.org/10.1086/670300

Maley, C. (forthcoming). Medium Independence and the Failure of the Mechanistic Account of Computation. *Ergo*.

Marr, D. (1982/2010). *Vision: A computational investigation into the human representation and processing of visual information*. MIT Press.

Milkowski, M. (2013). *Explaining the Computational Mind*, Cambridge, MA: MIT Press.

Piccinini, G., & Bahar, S. (2013). Neural Computation and the Computational Theory of Cognition: Cognitive Science. *Cognitive Science*, *37*(3), 453–488. https://doi.org/10.1111/cogs.12012

Piccinini, G. (2015). *Physical Computation: A Mechanistic Account*. Oxford University Press. https://doi.org/10.1093/acprof:oso/9780199658855.001.0001

Piccinini, G. (2020). *Neurocognitive Mechanisms: Explaining Biological Cognition* (1st ed.). Oxford University Press. https://doi.org/10.1093/oso/9780198866282.001.0001

Piccinini, G., & Craver, C. (2011). Integrating psychology and neuroscience: Functional analyses as mechanism sketches. *Synthese*, *183*(3), 283–311. https://doi.org/10.1007/s11229-011-9898-4

Polger, T. W., and L. A. Shapiro (forthcoming). “The Puzzling Resilience of Multiple Realization”. *Minds and Machines.*

Priorelli, M., & Stoianov, I. P. (2023). Flexible intentions: An Active Inference theory. *Frontiers in Computational Neuroscience*, *17*, 1128694. https://doi.org/10.3389/fncom.2023.1128694

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

Shagrir, O., & Bechtel, W. (2017). Marr's computational level and delineating phenomena. In D. M. Kaplan (Ed.), Explanation and integration in mind and brain science (pp. 190–214). Oxford University Press.

Shagrir, O. (2022). *The nature of physical computation*. Oxford University Press.

Sterling, P., & Laughlin, S. (2015). *Principles of Neural Design*. The MIT Press. https://doi.org/10.7551/mitpress/9780262028707.001.0001

Todesco, G. M. (2013). Cellular Automata: The Game of Life. In M. Emmer (Ed.), *Imagine Math 2: Between Culture and Mathematics* (pp. 231–243). Springer Milan. https://doi.org/10.1007/978-88-470-2889-0_25

Wajnerman Paz, A. (2017a). A mechanistic perspective on canonical neural computation. *Philosophical Psychology*, *30*(3), 213–234. https://doi.org/10.1080/09515089.2016.1271117

Wajnerman Paz, A. (2017b). Pluralistic Mechanism. *THEORIA*, *32*(2), 161–175. https://doi.org/10.1387/theoria.16877