Synthese (2022) 200:247
https://doi.org/10.1007/s11229-022-03725-8

ORIGINAL RESEARCH

Vertical-horizontal distinction in resolving the abstraction, hierarchy, and generality problems of the mechanistic account of physical computation

Jesse Kuokkanen[1]

Received: 25 May 2021 / Accepted: 27 April 2022 / Published online: 1 June 2022

Abstract

Descriptive abstraction means omission of information from descriptions of phenomena. In this paper, I introduce a distinction between vertical and horizontal descriptive abstraction. Vertical abstracts away levels of mechanism or organization, while horizontal abstracts away details within one level of organization. The distinction is implicit in parts of the literature, but it has received insufficient attention and gone mainly unnoticed. I suggest that the distinction can be used to clarify how computational descriptions are formed in some variants of the mechanistic account of physical computation. Furthermore, I suggest that, if this suggestion is adopted, it can be used to resolve what I call abstraction, hierarchy, and generality problems raised against mechanistic account of physical computation. According to the abstraction problem, the mechanistic account of physical computation is conceptually confused in claiming that physical systems process computational, abstract properties. An existing solution distinguishes between descriptive and metaphysical abstraction, suggesting that the abstraction problem unnecessarily postulates metaphysically abstract entities. The solution has been criticized for leading to what I call hierarchy and generality problems: it results in two separate hierarchies, one physical and one computational, making it problematic both to account for the generality of computational descriptions and to specify how the two hierarchies are related to each other. Adopting the vertical-horizontal distinction and the view that computational descriptions are achieved by horizontal abstraction allows one to account for the generality of computational descriptions, and to form a single hierarchy in which there are no separate hierarchies in need of integration.

Keywords Mechanistic account of computation · Vertical and horizontal abstraction · Abstraction problem · Hierarchy problem · Generality problem

Extended author information available on the last page of the article

1 Introduction

According to *the mechanistic account of physical computation* (MAC), physical computing systems process abstract and medium-independent, computational vehicles. A standard PC is such a physical computing system: it processes computational vehicles called *bits*. Bits are abstract and medium-independent since they are not defined according to their physical, realizing medium but according to their *degrees of freedom*. However, a traditional take in philosophy is that physical and abstract *contrast* with one another and cannot causally interact (Falguera et al., forthcoming). What gives?

Indeed, according to *the abstraction problem* (Hutto et al., 2019; Kersten, 2020), conceptual confusion plagues MAC's idea that physical systems process abstract properties. Kuokkanen & Rusanen (2018) have suggested a solution to the problem, according to which the abstraction problem makes an unnecessary assumption that abstract properties are *metaphysically abstract*, while MAC is talking about something else, namely *descriptive abstraction*. Descriptive abstraction means forming abstract descriptions by omitting information from the descriptions of phenomena. The more information the description omits, the more abstract it is. According to this suggestion, descriptive abstraction resolves the abstraction problem as it posits no metaphysically abstract entities.

Kersten (2020) has raised a concern concerning this suggestion: if abstract properties are not in the world but in our descriptions, objectively distinguishing between computational and non-computational systems becomes impossible. Pushing back and replying that abstract properties *are* a part of the world seems to reinvite the abstraction problem. One solution to this concern is to lean on what I call objective descriptive abstraction (ODA).[1] ODA means that some abstract descriptions are both objective and abstract. They 'denote a system's complex components, subsets of causal powers, and organizational relations that are operative (and thus explain a phenomenon) at one or more relevant levels of organization and produce a phenomenon with a suitable degree of generality' (Boone & Piccinini, 2016). In other words, they objectively track the world's structures without positing metaphysically abstract entities. This means that abstract properties need not literally be "in the world" to be objective.

However, Kersten (2020) argues that, due to what I call the *hierarchy* and *generality problems*, objective descriptive abstraction does not succeed in solving the problem. According to Kersten, endorsing ODA requires that MAC shows how the generality of a phenomenon and its abstract description match. Kersten argues that, in MAC, the generality of a phenomenon is determined according to its place in the implementational, physical hierarchy. However, according to the hierarchy problem, the relation between the implementational and computational mechanistic hierarchies in MAC is unclear (Elber-Dorozko & Shagrir, 2019a, b). If the relation between the generality-determining hierarchy and the computational hierarchy is unclear, ODA

[1] This is not a new idea, and it has been referred to in the literature using various terms: Boone & Piccinini (2016), for instance, write about the 'ontic role' that mechanistic abstraction plays in some cases, and Kersten (2020) talks about the 'ontic interpretation' of mechanistic abstraction.

faces a problem in determining the generality of the computational descriptions meant to describe physical phenomena. This is the generality problem: MAC lacks the means to match the generality of a phenomenon and its abstract description.

In this paper, I introduce a *vertical-horizontal distinction* in descriptive abstraction and examine it as a solution to the generality and hierarchy problems.[2] Vertical abstraction abstracts away levels of organization, while horizontal abstraction stays within one level of organization and abstracts away information from that level. I suggest that the horizontal-vertical distinction elucidates how computational descriptions are formulated in some variants of MAC. Furthermore, I suggest that those variants can use the vertical-horizontal distinction to address the problems mentioned above: first, objective descriptive abstraction resolves the abstraction problem. Furthermore, if computational descriptions are formed through horizontal descriptive abstraction, there are no separate hierarchies that need integration. Allowing one to specify the relationship between computational and physical descriptions, this also resolves the generality problem.

The idea of vertical-horizontal distinction is implicit in parts of the literature, but it has mostly gone unnoticed. Given the importance of the notion of abstraction for MAC, the possible implications of the distinction should be studied more closely. This article begins to fill the gap. I argue that the objective notion of abstraction should be kept separate from the epistemic and metaphysical notions. These clarifications throw light on the relation between MAC and abstraction.

As a caveat, the aim of this paper is to: (1) make explicit a distinction that is implicit in parts of the literature; (2) use it to illustrate how some variants of MAC treat computational descriptions; and (3) show how those variants can use the theoretical tools introduced to address some of the criticisms presented in the literature. I do not intend to argue for MAC, nor do I intend to argue for the general plausibility of the resulting theoretical framework.

I start by unpacking the basics of the mechanistic account of physical computation in the next chapter, which aids in digesting the problems of abstraction, hierarchy, and generality. Chapter 3 analyses abstraction: epistemic, subjective, objective, vertical and horizontal abstraction are introduced. Chapter 4 examines how MAC can use the conceptual toolkit developed to resolve the abstraction, generality, and hierarchy problems. Chapter 5 presents concluding remarks.

2 MAC: the mechanistic account of computation

The mechanistic account of computation (MAC) is one candidate among the theories of physical computation that seek to answer the question of what makes a physical system a computer. One of the alleged virtues of MAC is that it is an objective theory, providing a clear answer to the question of whether something is a computer or not, and ruling out paradigmatically non-computing systems such as rocks and walls.

[2] This idea is implicit, for instance, in Uskali Mäki's analysis (1992), although Mäki does not use the vertical-horizontal terminology for the distinction.

According to Gualtiero Piccinini's variant of MAC Piccinini, 2015, 2020; Ritchie & Piccinini, 2018), a physical system is a computer if it is a mechanism with the teleological function of manipulating abstract or medium-independent vehicles according to a rule. Computational vehicles, such as bits processed by standard computers, are medium-independent in the sense that they can be of any physical medium as long as they possess the proper structure and appropriate degrees of freedom. In the case of a bit, the crucial part is that it can sustain two stable and distinguishable states. It is not that important what a bit is made of or what kind of physical medium realizes it. According to MAC, a rock does not compute since it does not possess the teleological function of processing medium-independent vehicles.A mechanism 'is a structure performing a function in virtue of its component parts, component operations, and their organization' (Bechtel & Abrahamsen, 2005, 423). In other words, a mechanism can be explained by breaking it down or decomposing it into its constituent components or parts. These parts can be further decomposed into their constituent components, and so on. Borrowing an example from Craver (2007), we might be interested in how a neuron releases neurotransmitters. The mechanistic approach to explanation sees the target phenomenon as something brought about by its constituent parts, their structure, organization, and activities. In this case:

> 'The mechanism begins, we can say, when an action potential depolarizes the axon terminal and so opens voltage-sensitive calcium (Ca2+) channels in the neuronal membrane. Intracellular Ca2+concentrations rise, causing more Ca2+to bind to Ca2+/Calmodulin dependent kinase. The latter phosphorylates synapsin, which frees the transmitter-containing vesicle from the cytoskeleton. At this point, Rab3A and Rab3C target the freed vesicle to release sites in the membrane. Then v-SNARES (such as VAMP), which are incorporated into the vesicle membrane, bind to t-SNARES (such as syntaxin and SNAP-25), which are incorporated into the axon terminal membrane, thereby bringing the vesicle and the membrane next to one another. Finally, local influx of Ca2+at the active zone in the terminal leads this SNARE complex, either acting alone or in concert with other proteins, to open a fusion pore that spans the membrane to the synaptic cleft' (Craver, 2007, 5) (Fig. 1).

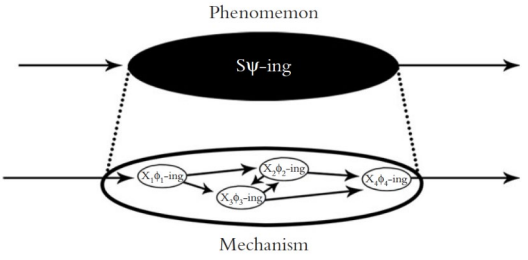

Fig. 1 An illustration of how a phenomenon (top) is explained by its mechanism (bottom). From Craver (2007, p. 7)

Figure 1.1. A phenomenon (top) and its mechanism (bottom).

MAC takes computational systems and explanations to be decompositional in this sense. However, note that in the example of a neuron above, the explanation describes physical and chemical, and thus *medium-dependent*, properties. When it comes to computing systems, mechanistic explanations are different in that they do not describe medium-dependent properties of the system or its components. Instead, they describe *computational or mathematical* parts and properties, which are medium-independent.

In Piccinini's theory (Piccinini, 2015), a computing system consists of primitive and complex computing components, and of components that do not themselves compute but still contribute to the overall functioning and computation of the system. Primitive computing components can be combined to build complex components, which can be further combined to build even more complex components and networks of components. In digital computers, the primitive computing components are called logic gates. When it comes to standard, everyday computers, logic gates are made of electrical transistors joined together. But, for something to be a logic gate, it does not really matter what it is made of: we can even build logic gates out of vacuum tubes or domino blocks. In this sense, the property of being a logic gate is *medium-independent*. By joining a number of logic gates at their extremities, we can build more complex computing components, such as Boolean circuits, which belong to a class of combinatorial computing components. Boolean circuits can be further combined to form even more complex computing components, such as Arithmetic Logic Units (ALU) (Piccinini, 2015).

MAC suggests that when explaining the operation of a computing component or a computing system, we decompose it into its constituent parts, their structure, and its organization. The resulting explanation in this case is mechanistic. Again, it is different from the earlier neuron example in the sense that explaining computational components does not appeal to the physical or chemical properties of the realizing physical medium: it describes computational and abstract, medium-independent parts and properties. The computational parts can be further decomposed and analysed into their computational parts, and so on, until we reach the primitive computing components.

Primitive computing components are primitive in the sense that their constituent parts do not perform computations, and hence, the operation of the primitive computing components cannot be further explained in computational terms. Primitive computing components can, however, still be analysed and explained mechanistically. At that point, the explanation is no longer computational because no computing components or computations can be found below that level. Instead, the explanation will describe medium-dependent properties.[3]

Note that the mechanistic framework entails a hierarchy of *levels*. The workings of an ALU in a digital computer can be decomposed into and explained by its component parts, Boolean circuits. The operation of the Boolean Circuits can be further decomposed into and explained by their component parts, logic gates. The ALU is at a *higher mechanistic level* than the Boolean circuits constituting it, which are at

[3] In a strict sense, then, it is the *implementation* of the primitive computing component that is analysed.

a higher mechanistic level than the logic gates composing them.[4] We will return to the idea and role of levels later when discussing the idea of vertical and horizontal abstraction.

3 Descriptive abstraction: roles, objectivity, direction

The term 'abstract' can mean different things. A traditional meaning in philosophy is metaphysical, meaning non-spatiotemporal or something that contrasts with concreteness. Metaphysically abstract objects and properties are also causally inefficacious: they do not have any causal powers and cannot manipulate or be manipulated by concrete entities. Even though there is no consensus regarding what makes something abstract in the metaphysical sense, the abstract-concrete distinction has been important in philosophy (Rosen, 2020). Some popular candidates for metaphysically abstract entities have been propositions, concepts, and mathematical entities.

A different way of using the term, popular in philosophy of science, is *descriptive abstraction*. In descriptive abstraction, one omits information from the description of a phenomenon. The more details omitted, the more abstract the description. Descriptive abstraction is ubiquitous in scientific explanation and modelling (Boone & Piccinini, 2016; Mäki, 1992; Piccinini, 2015; Portides, 2018; Raerinne, 2018; Tee, 2020; Weisberg, 2013). It can be both objective or non-objective and, as I suggest, vertical or horizontal. When I speak of 'abstract' or 'abstraction', I mean descriptive abstraction unless stated otherwise.

Boone & Piccinini (2016) discuss both 'ontic' and 'epistemic' *roles* of abstraction. Abstraction plays an *epistemic role*, for example, when we need to abstract away details simply because we do not know them. Another epistemic role for descriptive abstraction is the aim to reduce the complexity of the target phenomenon. Reducing the complexity is often important when we want to make complex phenomena more understandable and predictable. It can make computationally intractable problems tractable or bring phenomena within our reach that would otherwise require unrealistic amounts of modelling resources. Furthermore, scientists are often interested in some specific aspect of a larger phenomenon, which requires abstracting away, and isolating the target from, other aspects of the phenomenon.[5] It is likely that abstraction always serves some of these epistemic roles.

Descriptive abstraction can also be what Piccinini & Boone call *ontic*: some abstractions 'play the *ontic role* of identifying the specific complex components, sub-

[4] As Craver (2015) notes, there are various ways to cash out the levels metaphor: '[a] survey of kinds of levels drawn from science and philosophy would have to include levels of abstraction (Floridi, 2008), aggregation (Wimsatt, 1997), analysis (Churchland & Sejnowski, 1992; Shepherd, 1994), causation and explanation (Kim, 1998), implementation (Marr, 1982), organization (Churchland & Sejnowski, 1992), processing (Craik & Lockhart, 1972), realization (Gillett, 2002), sizes (Wimsatt, 1976), sciences, theories, and explanations (Oppenheim & Putnam, 1958).' Which one is the best candidate for cashing out the metaphor is not discussed in this paper, nor is the general feasibility of the levels metaphor itself (e.g., Eronen 2015).

[5] In scientific modelling, researchers also make isolations, idealizations, and simplifications (Mäki, 1992). Their relation to abstraction is beyond the scope of this paper.

sets of causal powers, and organizational relations that produce a suitably general phenomenon' (Boone & Piccinini, 2016).[6] In other words, some abstractions aim to track how things really are in the world. When such an abstraction succeeds and captures the structures it is aimed to capture, it is reasonable to say that the posited features of the abstraction are not just a part of the abstract description. In such a case, the abstraction tells us something real about its targets and, in this sense, the abstract features or properties are also a part of the world.

The roles of abstraction discussed by Boone & Piccinini are concerned with the pragmatic or intentional aspect of abstraction. However, it is worth noting that the abstraction might or might not track the structures of the world *regardless* of its role. For this reason, we should distinguish the abstraction's *objective status* from its role. When an abstraction successfully tracks and matches with the structures of the world, we are dealing with *objective descriptive abstraction*, regardless of its role.

We can also distinguish the *direction* of abstraction, given that one buys into the ubiquitous metaphor of levels. It is intuitive for us to organize and conceptualize the world in terms of levels of organization: electrons are at a lower level than molecules, which are at a lower level than synapses, which are at a lower level than brains. Even though the levels metaphor takes on different meanings depending on the context, it is also widely used in the sciences (Craver, 2015).[7] When trying to build a model of, say, the structure and function of a neuron, one option is to capture as much detail as possible at all levels. One might want to look at the components that build up or *constitute* the neuron, and whether the neuron itself is a part of some larger mechanism it contributes to. In such a case, the description can be said to span multiple levels. Another option is to focus on some very specific aspect of a phenomenon within one level of organization, in which case we restrict our investigation to that specific level.

In a hypothetical example where a model captures all the details from a single level but abstracts away other levels from the description, one performs only *vertical* abstraction: higher and lower levels are omitted and abstracted away. When we omit information and details *within* one level, in turn, we perform *horizontal* abstraction. In scientific practice, these usually go hand in hand: 'Obviously,' as Uskali Mäki put it, 'any instance of theory or model formation involves both kinds' (Mäki, 1992, 323).[8] Vertical and horizontal *directions* of abstraction are implicitly mentioned in parts of the literature but they are rarely explicated or given sufficient attention, Mäki's analysis being a refreshing exception.

Boone & Piccinini provide one example of the implicit use of the distinction when analysing mechanistic abstraction: 'two types of abstraction must be performed: (i) abstraction to sufficiently general types of components, properties, and organizational relations; and (ii) abstraction from lower levels of organization to higher levels of organization' (2016, 691). In the terminology I suggest, abstraction 'from lower

[6] While Boone & Piccinini distinguishes the ontic role from the epistemic roles, one might also argue that the ontic role is just one epistemic role among others.

[7] The discussion regarding the *plausibility* of the levels metaphor (Eronen, 2015; Fehr, 2004; Machamer & Sullivan, 2001; Piccinini, 2020; Thalos, 2013) is not discussed in this paper.

[8] Mäki (1992) talks about vertical and horizontal *isolation* and distinguishes isolation from abstraction. In Mäki's framework, abstraction is a subspecies of isolation. I do not discuss the relation between isolation and abstraction in this paper.

levels or organization to higher levels of organization' is vertical abstraction, and 'abstraction to sufficiently general types of components' can be seen as an example of horizontal abstraction. In the case of horizontal abstraction or abstraction to sufficiently general types of components, we lock in one level of organization of a system and focus on entities on that level. Then, we abstract away physical details from certain types of entities in order to acquire a general description of an entity type that captures all sufficiently similar entities. In this process, we do not move between different levels of organization. Instead, we stay within one level and, by abstracting away physical details from that level, we acquire sufficiently general descriptions of components.

4 Computational descriptions as horizontal abstractions in MAC

Advocates of MAC usually use the term 'abstract' to mean descriptive abstraction: the more details a description omits, the more abstract it is. According to Piccinini (2015), for example, we can describe a Dell Latitude laptop in several ways. We can easily describe it as a Dell Latitude. We could, if we wanted to, describe all its component parts instead. Being even more thorough, we could say basically the same thing in terms of the system's electrical circuits, or even atoms. This would be cumbersome and simply calling it a Dell Latitude saves us a lot of headaches.

According to Piccinini, computational and mathematical descriptions are similarly abstract: 'Mathematical descriptions of concrete physical systems,' Piccinini writes, 'are abstract in this sense. They express certain properties … while ignoring others … [C]omputational descriptions of concrete physical systems are mathematical and thus abstract in the same sense' (Piccinini, 2015, 9).

Mathematical or computational descriptions can be given for all sorts of objects and phenomena, such as weather, rocks, and food digestion. Many, Piccinini included, would say that this is not enough to make the described systems computational: for a physical system to count as a computer, it must meet additional criteria. According to MAC, a physical computing system is a *functional mechanism that has a teleological function of processing or manipulating medium-independent vehicles*. In other words, a physical computing system has a job to do, and that job is to perform computation or manipulate medium-independent or abstract vehicles according to certain rules (Piccinini, 2015).[9]

A vehicle is either (1) a variable, meaning a state that can take different values and change over time, or (2) a specific value of such a variable (Piccinini, 2015, 121). A *bit* is a familiar example of a computational, medium-independent vehicle. It is a variable defined by its degrees of freedom, being able to take one of two possible values–often labeled 0 and 1. A bit is medium-independent because it is defined by its degrees of freedom, not by its physical fingerprint or the implementing medium. A solid-state transistor is a commonly used for its implementation: it can be either on

[9] Rules need not be explicitly encoded or represented within the system: they can be mere descriptions of the system's rule-following behavior, such as mappings from inputs, and possibly internal states, to outputs.

or off, corresponding to the two possible values of a bit. However, a bit can also be implemented in other media, such as a vacuum tube or a domino block.

With these conceptual distinctions in place, we can see that Piccinini is ambiguous regarding the *direction* of abstraction. In his example concerning the Dell Laptop, he is talking about *vertical* abstraction: we abstract away its components and *lower levels of mechanism or organization*.[10] When he talks about mathematical descriptions, he does not say anything regarding the direction of abstraction. All Piccinini says is that computational descriptions are 'similar' to the Dell Latitude example as they omit some properties while including others. I suggest that in Piccinini's theory of mechanistic computation, the formulation of mathematical or computational descriptions is different from the Dell example. The Dell example employs vertical abstraction but formulating computational or mathematical descriptions in Piccinini's MAC crucially relies on *horizontal* abstraction.

In Piccinini's theory, a physical computing system is a mechanism: it can be decomposed into its constituent parts, and the properties at the higher level are explained by its components, their structure, and their activities at the lower level. There is a floor level in the computational hierarchy which cannot be further analysed computationally, since its constituent parts do not perform computations. Even though the floor level cannot be analysed computationally, it can still be analysed mechanistically. At that point, the explanation and analysis become *medium-dependent* in contrast to computational analysis, which is medium-independent.

The floor level of a computational hierarchy consists of *primitive computing components*. In standard, physical digital computers, primitive computing components are logic gates that are made by combining *transistors* with each other in a certain way. Depending on the electric current, the transistor's state is interpreted as either on or off, which is further interpreted as values or states 1 and 0. In Piccinini's account, the transistors are not themselves performing any computation, but they are the constituting components of the logic gates, which are the primitive computing components.

Even though transistors are not computational components in Piccinini's theory, the states of transistors form the inputs and outputs of logic gates. In other words, computational inputs and outputs are transistor states in standard digital computers. The value of a computational vehicle, in this case a bit, is determined by looking at the current state of a given transistor. In MAC, describing the state of a transistor as being either '1' or '0' means performing descriptive abstraction: what is important is the state of the transistor, not its other properties. Note that it is *horizontal abstraction* that does the heavy lifting: one omits other properties *at the same level of mechanism* and focuses on the electrical current, which determines the state of the transistor. The transistor's constitutive parts and its size, colour, or other properties are irrelevant.

This is also the case with logic gates. In standard physical digital computers, logic gates are made by combining transistors with each other in a certain way. In this sense, in standard digital computers logic gates are transistor networks. Using Craver's (2007) framework of levels of mechanisms, we can say that combining single

[10] Strictly speaking, probably all cases of abstraction involve both directions of abstraction, as mentioned in the Chap. 3.

transistors in such a way that they form a logic gate constitutes a new, higher level of mechanism, since the logic gate possesses causal powers and properties that a single transistor lacks. However, when we describe the transistor network through computational terminology as a logic gate, it is the abstraction happening at *that mechanistic level* which does the heavy lifting. Both vertical and horizontal abstraction are at work, but what does the crucial work in capturing the relevant properties for the computational description is the horizontal abstraction.

The same goes for more complex computing components. Once one has made the appropriate horizontal abstractions at different vertical levels, one arrives at a hierarchy of computational descriptions. In MAC, computational descriptions are located at a certain point on the horizontal axes. One arrives at the point of computational description by omitting irrelevant physical properties at each level. In this sense, computational explanation or analysis is *parallel* to the physical and medium-dependent mechanistic hierarchy. In Piccinini's MAC, each mechanistic level is *wide*: at one end, there is full physical detail. Moving towards the other end by performing horizontal abstraction, one arrives at computational descriptions at some point.

Summing up this section, the bottom line is that a mechanistic part-whole relation is a constitution relation. A physical-computational relation is an implementation relation. Constitution is not implementation. The case of a transistor implementing a medium-independent vehicle, such as a bit, is different from Piccinini's laptop example. Whereas the components of the Dell Laptop are *parts* of the computer, the state of a transistor is *not* a part of the vehicle. A transistor does not constitute a computational vehicle, but implements one. When Piccinini writes that computational descriptions are *abstract* in the same sense as the Dell example, he is talking about *descriptive abstraction* but is vague about the direction of the abstraction.

In the case of describing the states of a transistor as '0' or '1', one abstracts away physical details at the same level of organization. In the same sense, the physical structures that implement a logic gate are not its constituting parts: we do not move vertically from one level of organization to the next. Instead, we stay within one level of organization and abstract away physical details from that level. As a result, formulating computational description is, *to some extent*, 'similar' to the case of a Dell laptop as both are cases of descriptive abstraction. They are, however, different in the sense that Piccinini arrives at computational descriptions by performing *horizontal* abstraction, whereas choosing to describe the computer as a Dell rather than a collection of its component parts is a case of *vertical* abstraction.

As a potential objection, one might worry that horizontal and vertical abstraction are not easily separated.[11] Imagine a scenario where one decides to treat X2 and X3 as one component, X5, ignoring the finer-detail interactions between X2 and X3. Within this level, this seems to be a case of horizontal abstraction. But we can also imagine a separate scenario where X1, X4, and X5 are at one level and X2 and X3 are at a lower level than X5 since they constitute it. In this case, describing only X5 would seem like a vertical abstraction.

However, in case one simply decides to 'treat' X2 and X3 as one component, X5, it suggests that X5 is not actually a mechanism but rather a placeholder for an (X2

[11] Thanks to the anonymous reviewer for making this point.

& X3) aggregate.[12] In this case, X5 should not mechanistically analysed or decomposed. Second, if X5 *is* a mechanism and does the relevant work of (X2 & X3) in the mechanism at the same level, then it is X5 which should be in the picture, not X2 and X3: a mechanism and its parts are not on the same level. In this case, X2 and X3 are parts of X5 and might be described if one decides to analyse and decompose X5. Furthermore, if X5 is a mechanism constituted by X2 and X3 but is on a higher mechanistic level, then it should not be in the picture at the same level with X2 and X3.[13]

It might also be beneficial to briefly consider how the idea of a vertical hierarchy of horizontally wide mechanistic levels relates to Marr's (Marr, 1982) three levels of analysis of information-processing systems.[14] These are levels of: (1) computation; (2) representational and algorithm; and (3) implementation. In Marr's trichotomy, the computational level describes *what* is being computed and *why*. The level of representation and algorithm describes both the representational system used and the appropriate steps or the procedure for carrying out the computation in question. The implementational level tells us how the algorithm and the representation it uses can be realized physically.[15]

First, in the vertical hierarchy of wide mechanistic levels, the Marrian level of implementation can be thought of as corresponding to the medium-dependent end of each level: it retains the physical properties in the descriptions.[16] Second, Marr's level of representation and algorithm specifies the representational system used and the algorithms defined over the representations. In the case of a cash register, 'we might choose Arabic numerals for the representations, and for the algorithm we could follow the usual rules about adding the least significant digits first and "carrying" if the sum exceeds 9' (Marr, 1982, 23). Here, representations seem to lie at the medium-independent end of each mechanistic level, which is arrived at by horizontal descrip-

[12] This is related to a more foundational question of what mechanisms are and how they differ, say, from mere aggregates. Craver (2015), for example, writes that '[t]hrough aggregation or organization, wholes have causal powers that their parts individually do not have. An activity at a higher level of mechanistic organization is quite literally more than the sum of its parts. It is not an aggregate.' A more detailed discussion is, however, beyond the scope of this paper.

[13] A related concern (also raised by the anonymous reviewer) is that the horizontal location at the mechanistic level affects how that specific point or part unravels or decomposes: a horizontally more abstract formulation of a given phenomenon results in a different analysis from the analysis of the same phenomenon in a less abstract description. I feel that this touches on some fundamental questions regarding how to think about mechanisms and mechanistic hierarchies, and discussing them at sufficient length is beyond the scope of this paper. However, a brief reply would be that I do not see this as a problem. On the contrary, I think it seems intuitive and non-problematic that a mechanistic hierarchy can "branch", depending on the situation. Furthermore, the suggested framework and the resulting mechanistic hierarchy do not need to be interpreted as 'monolithic divisions in the furniture of the world' (Craver, 2015).

[14] Thanks to the anonymous referee for this suggestion.

[15] The descriptions provided here for each level are rough, and much more could be said about them. Indeed, the discussion revolving around Marr's idea forms a literature of its own and is beyond the scope of this paper. For more detailed discussion, see e.g., (Bechtel & Shagrir, 2015; Hardcastle & Hardcastle 2015; McClamrock, 1991; Peebles & Cooper, 2015; Rusanen & Lappi, 2016; Shagrir, 2010b; Shagrir & Bechtel 2018).

[16] Two caveats: first, I do not claim that Marr's three levels map neatly onto, or strictly correspond to, this so-called single hierarchy view of vertical hierarchy of wide mechanistic levels. Second, the idea that the implementational-level description is at the 'medium-dependent end' of a mechanistic level does not mean that Marr's implementational-level description should contain *all* physical detail.

tive abstraction–by omitting the implementational, physical or medium-dependent properties.

Marr's computational level specifies the mathematical function being computed and 'why the computation is appropriate for a given task under certain information processing constraints' (Rusanen & Lappi, 2016). As the function to be computed is specified in mathematical terms, it means that, horizontally, it is located at the medium-independent end of the hierarchy. Also, as the function by itself describes the behavior of the computational *system* at its higher organizational level and not its lower-level components, it is vertically located at the top of the hierarchy.[17]

5 Resolving abstraction, generality, and hierarchy problems

According to MAC, computational physical systems have a teleological function of manipulating abstract or medium-independent vehicles. The abstraction problem (Hutto et al., 2019) states that it is conceptually confused to claim that physical entities manipulate or causally interact with abstract or medium-independent entities, since abstract means non-concrete or non-spatiotemporal. However, one can understand the term 'abstract' in different ways. One option is to take a position according to which bits are *metaphysically* abstract entities and exist in the same way as platonic numbers. In this case, one would have to explain how is possible for physical entities to manipulate entities that are metaphysically abstract and causally impotent.

However, this is not the only option available. In MAC, computational vehicles are abstract in the sense that medium-dependent physical properties are abstracted away from the description (Kuokkanen & Rusanen, 2018; Piccinini, 2015, 2020). Medium-independent, computational vehicles result from *descriptive abstraction*. With descriptive abstraction, the abstraction problem does not arise in the first place. Computational vehicles are not abstract in the sense that they lack causal powers as metaphysically abstract entities do.

Kersten (2020) argues that descriptive abstraction leads to loss of objectivity in MAC. According to his argument, the fact that our *descriptions* are abstract entails that medium independence is 'not a property of the world' but of our descriptions. This problem can be resolved with the conceptual distinctions introduced: descriptive abstraction can be non-objective or objective. In *objective descriptive abstraction*, properties described by abstract descriptions can be seen as properties of the world, but they need not be metaphysically abstract. In other words, descriptive abstraction does not entail anything regarding the objectivity of the abstraction in question.

Kersten does consider the possibility of objective descriptive abstraction. He argues that it leads to what I call *hierarchy and generality problems*, and hence objective descriptive abstraction is not a viable option for solving the abstraction problem. These problems will be introduced and discussed next.

[17] This is a rough, preliminary idea concerning the relationship between Marr's levels and the framework developed here. Furthermore, this does not need to entail anything regarding the dependence or independence between levels, which is a discussion of its own (Bechtel & Shagrir, 2015; Eliasmith & Kolbeck, 2015; Kaplan, 2011; Piccinini, 2006; Shagrir, 2010a).

Kersten (2020) argues that for objective descriptive abstraction to work, and to ensure that the resulting abstract descriptions are not arbitrarily abstract but track the worldly features they aim to, one should have a method for establishing the generality of a phenomenon in question. In other words, one should have a method for showing that the abstract or computational description maps onto the phenomenon in the world it aims to track. One such strategy is called cross-situational stability (Boone & Piccinini, 2016).

In cross-situational stability, researchers look at several instances or situations of a given phenomenon and try to identify properties and mechanisms that are common to the phenomenon across the different situations. For example, researchers might study various instances of rat navigation tasks, trying to capture the general mechanism for rat navigation across different situations. Here one might find, for example, that the cognitive process of memory is one component in the higher-level phenomenon of navigation.

Kersten (2020) argues that in such a case, one has tools for establishing the generality of the phenomenon in question: rat navigation is more general than rat memory, because navigation takes place on a higher mechanistic level. In other words, according to Kersten *a phenomenon's generality is determined by its place in the vertical mechanistic hierarchy*: '[e]xplanatory shifts from higher-level phenomenon, such as rat navigation, to lower-level component and activities, such as rat memory, involve a reduction in the generality of the phenomenon being explained' (Kersten, 2020).

However, Kersten argues that there is a problem for MAC in relying on cross-situational stability to establish the generality of computational descriptions. In the example of rat navigation, the situations are described using physical or medium-dependent properties. Computational or mathematical descriptions, however, are abstract or medium-independent: it seems that there is a gap between the physical and mathematical descriptions.

The generality and hierarchy problems are intertwined in Kersten's argument: first, Kersten argues that MAC has a problem in determining the generality of abstract descriptions. According to Kersten, this is because in mechanistic explanation, the generality of a phenomenon is determined by its vertical placement within the physical, mechanistic hierarchy. One might reply by adopting the MAC stance, according to which computational explanations are also mechanistic, thus offering a similar mechanistic hierarchy for determining the generality of the description. In this case, however, one must show how the resulting abstract hierarchy is related to the physical one so that the abstract descriptions correspond correctly with the physical phenomena. The framework introduced in this paper provides additional clarification and one potential solution for MAC to the generality and abstraction problems.

First, we can point out that Kersten seems to equate the generality of a phenomenon with its mechanistic *vertical* level. However, just as there seems to be both horizontal and vertical abstraction, so there appear to be two different kinds of generality resulting from these different kinds of abstraction.[18] One kind of generality results from vertical abstraction: here, one abstracts away internal processes and composition at the lower levels. The resulting higher-level description of phenomenon is

[18] Thanks to the anonymous referees for stressing this point.

compatible with a wider range of lower-level processes and increases generality in this sense. Horizontal abstraction results in a different kind of generality, in which the physical properties at one mechanistic level are omitted, and computational or medium-independent descriptions are formed. This is how Piccinini, for example, seems to write about the relationship between physical and computational or mathematical descriptions in MAC.[19]

The idea behind Kersten's generality argument is that MAC is unable to determine the generality of computational descriptions because computational descriptions are detached from their implementational properties and there is no way of matching the two or tracking the computational properties back to their implementational counterparts. In other words, the implementational and computational hierarchies are not integrated. The framework and approach suggested for MAC in this paper solves this problem: computational properties are not detached but arrived at simply by descriptive abstraction, resulting in a single mechanistic hierarchy that includes both implementational and computational properties.

This relates to and has some implications for MAC on what I call the hierarchy question concerning implementational and computational properties (Elber-Dorozko & Shagrir, 2019b). In the separate hierarchy view, implementational and computational properties are kept separate in their own hierarchies and bridged via an implementation relation when there is a proper mapping between the two. In the single hierarchy view, there are no separate hierarchies; instead, the implementational and computational properties sit together in one mechanistic hierarchy. The observations and ideas presented in this paper suggests that MAC opts for a single hierarchy view. This is because a central motivation for the separate hierarchy is the claim that mechanistic and computational hierarchies do not always systematically match or integrate. If the observations and ideas presented in this paper are correct, this is in contrast with MAC. Furthermore, Elber-Dorozko & Shagrir (2019b) take the single hierarchy view simply as 'lumping' the computational and physical properties together in each level. The framework developed in this paper clarifies the idea of the single hierarchy view, proposing one more detailed way to think about the relationship between computational and physical descriptions in MAC.[20]

Summing up, we can start by noting that objective descriptive abstraction resolves the abstraction problem. Abstract, computational descriptions can be objective without entailing metaphysically abstract entities. The vertical-horizontal distinction can be used to resolve the generality problem: in MAC, computational or mathematical descriptions are arrived at by horizontal abstraction in the same mechanistic hierarchy one uses for determining the generality of a physical phenomenon. Furthermore, one should note that both kinds of abstraction result in different kinds of generality. This relationship between physical and computational or mathematical properties results in a single hierarchy view, in the mechanistic of which each level in the hierarchy is wide, varying in its amount of physical detail depending on the amount

[19] However, one can adopt the vertical-horizontal framework of mechanistic levels and abstraction regardless of whether one thinks that computational or mathematical descriptions are arrived at by horizontal abstraction.

[20] However, I do not intend to argue for or against either the single or separate hierarchy view.

of horizontal abstraction. This also solves the hierarchy problem: there are not two separate hierarchies needing integration.

6 Conclusions

In this paper, I have analysed the notion of abstraction and distinguished between metaphysical and descriptive abstraction, the role and objective status of descriptive abstraction, and the vertical and horizontal directions of descriptive abstraction.

Descriptive abstraction is omission of information from the descriptions of phenomena. It does not entail positing metaphysically abstract entities. Descriptive abstraction plays several epistemic roles. One role for descriptive abstraction is ontic, in which case the aim is to form objective abstract descriptions. While the abstraction's epistemic role is concerned with the pragmatic or intentional aspect of the researcher, the *objective status* of descriptive abstraction can be objective or non-objective regardless of its epistemic role. The objective status of the abstraction depends on whether it successfully captures the structures of the world.

Vertical descriptive abstraction happens when we omit levels of organization or mechanisms from our descriptions. Horizontal descriptive abstraction happens when we omit details from our descriptions within one mechanistic level. I have suggested that in Gualtiero Piccinini's variant of mechanistic account of physical computation, computational or mathematical descriptions are arrived at by horizontal abstraction. I have also suggested that the distinctions introduced in combination with the aforementioned framing of the relationship between implementational and mathematical or computational descriptions can answer the so-called abstraction, hierarchy, and generality problems presented against the mechanistic account of physical computation.

According to the abstraction problem, it is conceptually confused to claim that physical systems can process abstract, computational vehicles. However, if we take computational descriptions as descriptive abstractions, the abstraction problem does not arise. Furthermore, the resulting abstract descriptions can be either non-objective or objective. According to the hierarchy problem, it is unclear how the physical and computational mechanistic hierarchies fit together. The solution sketched in this paper is that in Piccinini's variant of MAC, there is only one mechanistic hierarchy. Levels in this hierarchy are wide, varying in the amount of horizontal abstraction from implementational to computational. As a result, the problem of integrating two distinct hierarchies is resolved.

According to the generality problem, objective descriptive abstraction needs a way of establishing that the generality of computationally abstract descriptions matches with their target phenomena: if the generality of a phenomenon is determined by its vertical position within the implementational mechanistic hierarchy, as in Kersten's cross-situational stability example of rat navigation, it is unclear how the abstract or computational descriptions fit in. According to the suggestion sketched in this paper, in MAC computational descriptions are arrived at by performing horizontal abstraction on the mechanistic hierarchy. This addresses how computational descriptions relate to physical descriptions, providing the tools to resolve the generality problem. It also clarifies the notion of generality, suggesting that vertical and horizontal

abstraction result in different kinds of generality. This idea results in a single vertical hierarchy with wide mechanistic levels, also resolving the hierarchy problem: if there is a single hierarchy with wide mechanistic levels, the problem of relating two separate hierarchies to each other does not arise.

The idea of vertical and horizontal abstraction is implicit in parts of the literature, but has mostly gone unspecified and thus received insufficient attention. Given the importance of the notion of abstraction in mechanistic accounts of computation, the distinction and its implications should be studied further.

Acknowledgements I'm extremely grateful to the anonymous reviewers for their valuable comments and feedback. Also, I would like to express my deepest thanks to Anna-Mari Rusanen and Otto Lappi for all the discussions, feedback, comments, ideas, and general support along the writing process.

Authors' contributions not applicable.

Funding University of Helsinki.

Open Access funding provided by University of Helsinki including Helsinki University Central Hospital.

Availability of data and material not applicable.

Code Availability not applicable.

Declarations

Conflicts of interest/competing interests the authors have no financial or proprietary interests in any material discussed in this article.

Ethics approval not applicable.

Consent to participate not applicable.

Consent for publication not applicable.

References

Bechtel, W., & Abrahamsen, A. (2005). Explanation: a mechanist alternative. *Studies in History and Philosophy of Science Part C: Studies*. *History and Philosophy of Biological and Biomedical Sciences*, 36(2), 421–441. https://doi.org/10.1016/j.shpsc.2005.03.010

Bechtel, W., & Shagrir, O. (2015). The non-redundant contributions of Marr's three levels of analysis for explaining information-processing mechanisms. *Topics in Cognitive Science*, 7(2), 312–322. https://doi.org/10.1111/tops.12141

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

Churchland, P. S., & Sejnowski, T. J. (1992). *The computational brain*. MIT Press

Craik, F. I. M., & Lockhart, R. S. (1972). Levels of Processing: A Framework for Memory Research. *Journal of Verbal Learning and Verbal Behavior*, 11, 671–684

Craver, C. F. (2007). *Explaining the Brain*. Oxford University Press

Craver, C. F. (2015). Levels. In T. Metzinger & J. M. Windt (Eds.), *Open MIND* (Vol. 8). MIND Group. https://doi.org/10.15502/9783958570498

Elber-Dorozko, L., & Shagrir, O. (2019a). Computation and levels in the cognitive and neural sciences. In M. Sprevak & M. Colombo (Eds.), *The Routledge Handbook of the Computational Mind* (pp. 205–222). Routledge. https://doi.org/10.4324/9781315643670-16

Elber-Dorozko, L., & Shagrir, O. (2019b). Integrating computation into the mechanistic hierarchy in the cognitive and neural sciences. *Synthese*. https://doi.org/10.1007/s11229-019-02230-9

Eliasmith, C., & Kolbeck, C. (2015). Marr's Attacks: On Reductionism and Vagueness. *Topics in Cognitive Science*, 7(2), 323–335. https://doi.org/10.1111/tops.12133

Eronen, M. I. (2015). Levels of organization: a deflationary account. *Biology and Philosophy*, 30(1), 39–58. https://doi.org/10.1007/s10539-014-9461-z

Falguera, J., Martínez-Vidal, C., & Rose, G. (n.d.). Abstract Objects. In E. N. Zalta (Ed.), *The Stanford Encyclopedia of Philosophy* (Winter 202)

Fehr, C. (2004). Feminism and Science: Mechanism Without Reductionism. *Women's Studies Association Journal*, 16(1), 136–156

Floridi, L. (2008). The method of levels of abstraction. *Minds and Machines*, 18(3), 303–329. https://doi.org/10.1007/s11023-008-9113-7

Gillett, C. (2002). The dimensions of realization: a critique of the Standard view. *Analysis*, 62(276), 316–323. https://doi.org/10.1111/1467-8284.00377

Hardcastle, V. G., & Hardcastle, K. (2015). Marr's Levels Revisited: Understanding How Brains Break. *Topics in Cognitive Science*, 7, 259–273. https://doi.org/10.1111/tops.12130

Hutto, D. D., Myin, E., Peeters, A., & Zahnoun, F. (2019). The Cognitive Basis of Computation: Putting Computation in its place. In M. Sprevak, & M. Colombo (Eds.), *The Routledge Handbook of the Computational Mind* (pp. 272–282). Routledge

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

Kersten, L. (2020). How to be concrete: mechanistic computation and the abstraction problem. *Philosophical Explorations*, 23(3), 251–266. https://doi.org/10.1080/13869795.2020.1799664

Kim, J. (1998). *Mind in a physical world*. MIT Press

Kuokkanen, J., & Rusanen, A. M. (2018). Making too many enemies: Hutto and Myin's attack on computationalism. *Philosophical Explorations*, 21(2), 282–294. https://doi.org/10.1080/13869795.2018.1477980

Machamer, P., & Sullivan, J. (2001). *Leveling reduction*. University of Pittsburgh Philosophy of Science Archive

Mäki, U. (1992). On the Method of Isolation in Economics. *Poznan Studies in the Philosophy of the Sciences and the Humanities*, 26(4), 317–351

Marr, D. (1982). *Vision: a computational investigation into the human representation and processing of visual information*. Freeman Press

McClamrock, R. (1991). Marr's three levels: A re-evaluation. *Minds and Machines*, 1(2), 185–196. https://doi.org/10.1007/BF00361036

Oppenheim, P., & Putnam, H. (1958). Unity of science as a working hypothesis. In H. Feigl, M. Scriven, & G. Maxwell (Eds.), *Concepts, theories, and the mind-body problem, Minnesota studies in the philosophy of science II* (pp. 3–36). University of Minnesota Press

Peebles, D., & Cooper, R. P. (2015). Thirty Years After Marr's Vision: Levels of Analysis in Cognitive Science. *Topics in Cognitive Science*, 7, 187–190. https://doi.org/10.1111/tops.12137

Piccinini, G. (2006). Computational explanation and mechanistic explanation of mind. In M. DeCaro, F. Ferretti, & M. Marraffa (Eds.), *Cartographies of the mind: The interface between philosophy and cognitive science*. Kluwer

Piccinini, G. (2015). *Physical Computation: A Mechanistic Account*. Oxford University Press

Piccinini, G. (2020). Neurocognitive Mechanisms: Explaining Biological Cognition. In *Journal of Chemical Information and Modeling* (Vol. 53, Issue 9). Oxford University Press
Portides, D. (2018). Idealization and abstraction in scientific modeling. *Synthese*. https://doi.org/10.1007/s11229-018-01919-7
Raerinne, J. (2018). Abstraction in ecology: reductionism and holism as complementary heuristics. *European Journal for Philosophy of Science*, 8(3), 395–416. https://doi.org/10.1007/s13194-017-0191-3
Rosen, G. (2020). Abstract Objects. In E. N. Zalta (Ed.), *Stanford Encyclopedia of Philosophy* (Spring2020 ed.). https://plato.stanford.edu/archives/spr2020/entries/abstract-objects/
Rusanen, A. M., & Lappi, O. (2016). On computational explanations. *Synthese*, 193(12), 3931–3949. https://doi.org/10.1007/s11229-016-1101-5
Shagrir, O. (2010a). Brains as analog-model computers. *Studies in History and Philosophy of Science Part A*, 41(3), 271–279. https://doi.org/10.1016/j.shpsa.2010.07.007
Shagrir, O. (2010b). Marr on computational-level theories. *Philosophy of Science*, 77(4), 477–500. https://doi.org/10.1086/656005
Shagrir, O., & Bechtel, W. (2018). Marr's computational level and delineating phenomena. *Explanation and Integration in Mind and Brain Science*, *1988*, 190–214
Shepherd, G. (1994). *Neurobiology*. Oxford University Press
Tee, S. H. (2020). Abstraction as an Autonomous Process in Scientific Modeling. *Philosophia (United States)*, 48(2), 789–801. https://doi.org/10.1007/s11406-019-00092-6
Thalos, M. (2013). *Without hierarchies: the scale freedom of the universe*. Oxford University Press
Weisberg, M. (2013). *Simulation and Similarity: Using Models to Understand the World.pdf*. Oxford University Press
Wimsatt, W. C. (1976). Reductionism, levels of organization, and the mind-body problem. In G. Globus, I. Savodnik, & G. Maxwell (Eds.), *Consciousness and the Brain* (p. 199). Plenum Press. https://doi.org/10.1007/978-1-4684-2196-5_9
Wimsatt, W. C. (1997). Aggregativity: Reductive heuristics for finding emergence. *Philosophy of Science*, 64(4), 372–384. https://doi.org/10.1086/392615
Ritchie, B. J., & Piccinini, G. (2018). Computational implementation. In Sprevak, M., & Colombo, M. (Eds.), The Routledge Handbook of the Computational Mind (Ch. 14). London: Routledge

Authors and Affiliations

Jesse Kuokkanen[1]

✉ Jesse Kuokkanen
jesse.kuokkanen@helsinki.fi

[1] University of Helsinki Faculty of Arts, Helsingin Yliopisto Humanistinen tiedekunta, Helsinki, Finland