The Role of the Environment in Computational Explanations

Abstract

The "mechanistic view of computation" contends that computational explanations are mechanistic explanations. Mechanists, however, disagree about the precise role that the environment – or the so called "contextual level" – plays for computational (mechanistic) explanations. We advance here two claims: (i) Contextual factors essentially determine the computational identity of a computing system (computational externalism); this means that specifying the "intrinsic" mechanism is not sufficient to fix the computational identity of the system. (ii) It is not necessary to specify the causal-mechanistic interaction between the system and its context in order to offer a complete and adequate computational explanation. While the first claim has been discussed before, the second has been practically ignored. After supporting these claims, we discuss the implications of our contextualist view for the mechanistic view of computational explanation. Our aim is to show that some versions of the mechanistic view are consistent with the contextualist view, whilst others are not.

1 Introduction

At the heart of the so-called "mechanistic view of computation" lies the idea that computational explanations are mechanistic explanations. Mechanists, however, disagree about the precise role that the environment – also called the "contextual level" (Miłkowski 2013) – plays for computational (mechanistic) explanations.

Some mechanists argue that contextual factors do not affect the computational identity of a computing system. For instance, Dewhurst suggests that computational processes are "fully captured by the physical structure of the computing mechanisms" (2018a, 111). Coelho Mollo emphasizes that "computational individuation takes place at the functional level" (2018, 3494). If, as he says, the functional profile of a system is a "mapping of input equivalence classes of physical states to output equivalence classes of physical states" (3494-5), then contextual aspects play no role for the computational identity of a system.[1]

Some mechanists agree that contextual factors affect the computational identity of the system (e.g., Piccinini 2008; 2015)[2], but they are less clear about the implication of this externalism for computational explanations. Does their position imply that computational explanations must refer to the mechanism that relates the system to its environment? It seems that this would be the case if we accept Kaplan's dictum that "computational models possess explanatory force to the extent that they describe the mechanisms responsible for producing a given phenomenon." (2011, 339).[3] Piccinini might have accepted this view when saying that "I find a wide (non-individualistic) construal of mechanistic explanation more plausible. … [And, i]n order to know which intrinsic properties of mechanisms are functionally [=computationally] relevant, it may be necessary to consider the interaction between mechanisms and their contexts." (Piccinini 2008, 219-20). We will return to consider various positions of the mechanistic view in section 5.

In this paper, we advance two claims about computational individuation and computational explanation. We argue that (i) contextual factors affect the computational identity of a computing system. Or in other words, individuating the "intrinsic" mechanism is not sufficient to fix what the system computes, but certain contextual factors play an essential role for the computational identity of a system. Secondly, we argue (ii) that it is not necessary to specify the causal-mechanistic interaction between the system and its context in order to offer a full computational explanation of a computing system.

[1] Coelho Mollo (forthcoming), however, also considers individuating computation by appeal to teleological functions.

[2] Piccinini claims that "[i]n order to know which of the computations that are implemented by a computing mechanism is explanatory in a context, we need to know the relevant relations between computations and contexts. Therefore, we cannot determine which computation is explanatory within a context without looking outside the mechanism. (…) Computations have effects on, and are affected by, their context." (2008, 231). See also Bechtel (2009) who argues that mechanistic explanations, in general, should look around, and Miłkowski (2017) who contends that environmental factors often constrain the functions that the mechanism computes.

[3] We note, however, that Kaplan might be an internalist about computation as he also writes that "[c]omputational models must describe the real structure of the computational mechanisms underlying the input–output mapping in order to explain." (368). If the mechanisms underlying the input-output mapping are all internal, then there is no need to appeal to contextual factors at all.

The two claims are related, but the relations between them is in no way obvious. Claim (i) is about individuation which is a metaphysical relation between states or properties and worlds. Claim (ii) is about explanations, which we take to be, roughly, epistemic models and descriptions developed by scientists in order to structure, understand, predict, and handle the world. When put in the context of computation, claim (i) is about the metaphysics of computation, namely, about the individuation conditions of computational states and properties. Claim (ii), in contrast, is about the requirements and contents of computational explanations. It is natural to think that the metaphysics has at least something to do with the adequacy of epistemically valuable explanations. Indeed, we conjecture that if computational individuation partially depends on contextual features (claim (i)), it is reasonable to assume that adequate computational explanations ought to refer to these contextual features as well. Our aim, however, is not to offer an explicit defense of this implication. Instead, our focus will be on arguing that, whilst computational explanations refer to external factors, they need not, and often do not, specify the causal-mechanistic interaction between the system and these external factors (claim (ii)).

The paper proceeds as follows. Section 2 focuses on the defense of claim (i). It introduces the notion of an automaton, and it argues that complex systems typically implement a number of inconsistent automata all at the same time. The challenge is to single out those automata of a system that correspond to its actual computations (given that the system computes anything at all), which cannot be achieved on the basis of the intrinsic features of the system alone. This establishes an externalist view of computation. Section 3 investigates whether extending the basis by including the immediate or close environment of computing systems does the trick, but argues that this extension still underdetermines the actual computations of a system. This means that we have to adopt a “long-arm” externalism about computation. Section 4 focuses on claim (ii) and argues that various different input mechanisms can be correlated with the same computations, and that it is not always necessary to specify the environment-to-system mechanism in order to explain a system’s computations. In Section 5 we discuss the implications of our contextual account for the mechanistic view of computation. Our aim is to show that some versions of the mechanistic view of computation are consistent with the contextual position, whilst others are not. Section 6 summarizes the paper and points to some open questions in the debate.

2 Automata as Computations

Whatever else physical computations may be, it is widely agreed that they are implementations of formal structures, also called "automata", by physical systems (cf. Hopcroft et al. 2000; see also Chalmers 1994, 392; Piccinini 2015, 10; 2017).[4] Automata are sets of formal states combined with rules specifying the manipulation of computational units and the transition between states. For instance, consider the following two simple Turing machines represented a) by the two quadruples <s1,1,0,s2> and <s2,0,1,s1> ("If in state 1 and scanning a stroke, erase it and go to state 2"; "If in state 2 and scanning a 0, print a stroke and go to state 1") and b) by the three quadruples <s1,1,R,s2>, <s2,1,L,s1> and <s2,0,L,s1> ("If in state 1 and scanning a stroke, move to the right and go to state 2"; "If in state 2 and scanning a 1, move to the left and go to state 1", "If in state 2 and scanning a 0, move to the left and go to state 1"). Given the convention that a Turing machine always starts on the leftmost stroke of the tape, the first machine forever erases and prints 1s and 0s alternately without ever moving on the tape. The second machine forever moves between two cells back and forth.

Both machines are automata in the above-mentioned formal sense. The first entry of each command quadruple represents a formal state, whereas the second and third entry of each quadruple represent rules specifying the manipulation of computational units. The fourth entry indicates the states into which the system transitions after having implemented the manipulations and/or movements. More complex commands specified in programming languages such as Python, C++, and R maintain this general structure in the sense that they always explicate a set of formal states, a set of rules for the manipulation of computational units and for the transition between states.

[4] Some authors have disagreed whether the implementation already requires semantics or not (cf. Sprevak 2010, Block 1990). For our argument, we presuppose that the implementation of an automaton by a system does not by itself have a semantic dimension.

The “computational level” of a given system then comprises at least a set of implemented automata along with the abstract function(s) realized by the system. A system implements an automaton if there is a mapping from states and state transitions of the automaton onto physical states and state transitions of the physical system. Most accounts of computational implementation add additional constraints such as counterfactual, causal, or even more pragmatic constraints (cf. Sprevak 2018 for an overview). It is controversial whether this definition fits all computing systems as, for instance, some neural networks do not seem to allow for separate states and discrete units. However, as long as “rules” are understood minimally in the sense of “regularities or mappings”, and if “states” and “computational units” are allowed to be continuous as well, the definition captures most of what today are considered computing systems.

Whilst most philosophers of cognitive science would agree that computations are at least implemented automata, it is controversial whether the implemented automata are already all there is to a computation. The question is especially pressing once it is acknowledged that computational systems often implement many different and inconsistent automata at the same time and place. This arguably shows that the intrinsic or local properties of the system do not suffice to determine the computation that the system performs.

Shagrir (2001) and Sprevak (2010) have developed two different but related examples showing precisely this. Shagrir has offered the example of a tri-stable electro-mechanical system P consisting of two gates: gate a and gate b. The system receives exactly three kinds of inputs to the gates, namely 1) 0-2.5V, 2) 2.5-5V, 3) 5-10V, but at any moment always the same combination of inputs for both gates. The output of each gate is listed in Table 1.

Input 1	**Input 2**	**Output gate a**	**Output gate b**
0-2.5V	0-2.5V	0-2.5V	0-2.5V
0-2.5V	2.5-5V	2.5-5V	2.5-5V
0-2.5V	5-10V	2.5-5V	5-10V
2.5-5V	0-2.5V	2.5-5V	2.5-5V
2.5-5V	2.5-5V	2.5-5V	2.5-5V
2.5-5V	5-10V	2.5-5V	5-10V

5-10V	0-2.5V	2.5-5V	5-10V
5-10V	2.5-5V	2.5-5V	5-10V
5-10V	5-10V	5-10V	2.5-5V

Table 1: System P (Shagrir 2001)

One may ask which computations these two gates perform. One answer is that, under the symbolic assignment of “0”= 0-2.5V and “1”=2,5-10V, both gates a and b are OR-gates (see Table 2).

Input 1	**Input 2**	**Output gate a**	**Output gate b**
0	0	0	0
0	1	1	1
0	1	1	1
1	0	1	1
1	1	1	1
1	1	1	1
1	0	1	1
1	1	1	1
1	1	1	1

Table 2: “0”= 0-2.5V and “1”=2,5-10V

However, under the symbolic assignment of “0”= 0-5V and “1”=5-10V, gate a becomes an AND-gate, whilst gate b becomes an XOR-gate (see Table 3).

Input a	**Input b**	**Output gate a**	**Output gate b**
0	0	0	0
0	0	0	0
0	1	0	1
0	0	0	0
0	0	0	0

0	1	0	1
1	0	0	1
1	0	0	1
1	1	1	0

Table 3: “0”= 0-5V and “1”=5-10V

In other words, depending on where the threshold is set at which an input represents a “1” instead of “0”, gates a and b implement different syntactic structures or automata. These automata are incompatible in the sense that, from a logical point of view, an AND-gate is a different thing than an XOR-gate.

Sprevak (2010) has reached a similar conclusion by examining a single gate which receives inputs of 0V and 5V and emits 0V and 5V the way listed by table 4.

Input 1	**Input 2**	**Output**
5V	5V	5V
5V	0V	0V
0V	5V	0V
0V	0V	0V

Table 4: AND-gate or OR-gate?

Under the symbolic assignment “0”=0V and “1”=5V, Sprevak’s electrical gate implements a syntactical AND-gate. However, under the symbolic assignment “0”=5V and “1”=0V, Sprevak’s electrical gate implements a syntactical OR-gate. Both examples suggest that the mechanisms of the two kinds of systems do not give any preference to any one of the incompatible syntactic structures or automata.

To see how the same observation can be made also for naturally computing systems, consider the brain of an aircraft marshaller. Human aircraft marshallers can be treated as highly complex computing systems. Their general task is to communicate specific signals to pilots. There is a system of internationally standardized and commonly known marshalling signals to that end, all of which have specific meanings. In order to fulfill their task, aircraft marshallers have to compute information by the tower as well as visual information about aircraft positions and sizes, ground conditions, and signal conventions in order to communicate the correct signals to the pilots and to guide the aircraft to the right position. For our current purposes, we simplify the complex task set performed and computed by an aircraft marshaller by focusing on the signal “place yourself facing me”, which is expressed by lifting both hands up (cf. figure 1).

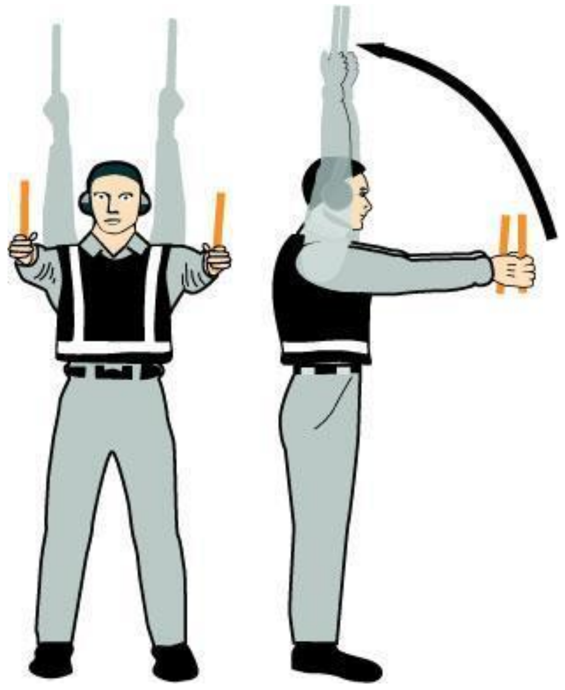
Figure 1: aircraft marshalling “Place yourself facing me!”

Suppose, for simplicity, that the aircraft marshaller receives signals from the airport tower in the form of beeps into his earphones. When he hears two simultaneous beeps on each side, he stretches the arms front and then lifts them up as illustrated in figure 1. When he hears a beep on one side only, he merely lifts one arm up. If the marshaller’s inputs and outputs are interpreted as follows:

- beep = “1”, no beep = “0”,
- both hands up = “1” =, one hand up or less = “0”,

…then the marshaller functions as a simple AND-gate. In contrast, if the marshaller's inputs and outputs are interpreted as follows:

- beep = "1", no beep = "0",
- at least one hand up = "1", no hands up = "0",

…then the marshaller functions as an OR-gate. This simple example shows that also natural computing systems can implement several incompatible automata all at the same time. It is only to be expected that the more complex a natural system, the larger the number of automata implemented by the system.

One may wonder which of these syntactic gates corresponds to the computation of a system. One interpretation could be to say that all automata realized by a system are computations of the system. For some versions of the intrinsic view of computation, this seems a natural conclusion. These versions essentially claim that the physical mechanism intrinsic to a system wholly determines the computations performed by the system (against claim (i) above). The conclusion implied by this assumption is that the intrinsic or local properties of some systems determine a great many such computations, all of which are performed in the same time and place (cf. Chalmers 1994, 397).

There are at least three problems with this conclusion: First, it is far removed from actual science as well as everyday attributive practice. Cognitive scientists do not usually consider computing systems as performing lots of incompatible computations all at the same time. Take David Marr (1982) as an example. Marr noticed that the visual system could detect edges by computing the extreme points of first-derivative operators, the second-order directional derivatives and perhaps other appropriate functions. He says that the system computes the function optimizing the "economy of computation" (p. 56), namely, the second-derivative of the Laplacian which is efficient relative to the first-derivative. Therefore, it is the one that is actually computed.

A similar approach seems to apply in non-scientific contexts. It does not matter whether the system in question is a cashier in a supermarket, Donald Trump's brain, or Deep Blue. We usually take the cashier to perform addition (and not subtraction at the same time), we take Donald Trump's brain to process the thought that James B. Comey lied (and not at the same time the thought that Comey told the truth), and we take Deep Blue to play chess (and not at the same time primitive versions of Go). In short, the assignment of computations to systems seems to be highly selective even in everyday contexts.

Second, such a conclusion renders the notion of computational equivalence implausible. Imagine that Donald Trump has a perfect molecule-by-molecule replica that inhabits TwinEarth, which itself is a global replica of Earth. Imagine further that Trump's replica contains a minimal divergence. A single neuron in the replica's brain is slightly different from its original: Instead of being bi-stable, it is tri-stable (cf. system P above). If all implemented automata correspond to computations of a system, then Real Donald and Replica Donald have a different "basic" computational structure and are not computationally equivalent. But if this the case, then every two systems that have a slight physical difference might be rendered as computationally different. The problem becomes more extreme if we look beyond digital computers. Analog computers allow for carving up the physically processed values in an infinite number of ways. In consequence, it becomes pointless to classify any two systems under the same computational type, or to describe them as computationally equivalent. Uncovering the "basic" computational structure of any given system then amounts to an extremely complex task. And since virtually no two systems in this world are entirely physically alike, almost no two systems are specifiable as computationally equivalent. Only very simple systems will turn out as performing exactly the same computations. However, such a conclusion clashes with scientific practice and even colloquial classification (cf. Shagrir 2019).

Third, if the implementation of automata by intrinsic properties is all there is to computation, extremely many systems will turn out to be computers. Many water taps, for instance, will turn out to be computing systems, as they implement an OR-gate under the following assignment: Input a: water = "1", no water = "0"; input b: water = "1", no water = "0", output: water = "1", otherwise = "0". Even hurricanes, floods, and seasons can be assigned various kinds of automata. However, characterizing all of these systems as computers would lead to limited pancomputationalism, a view that even according to many mechanists can hardly be maintained (Piccinini 2015, chap. 4).[5]

What these unpalatable implications of the intrinsic view of computations suggest is that there is more to computations than the automaton and the intrinsic properties. Contextual factors play an essential role for the computational identity of a computing system (cf. claim (i) above).

3 Close System-Context Interactions

One may concede the conclusion of section 2 and agree that contextual factors play an essential role for the computational identity of a computing system, and that its intrinsic properties do not wholly determine a system's computational identity. However, in order to save a version of the intrinsic view, one could argue for a milder position, according to which the system in conjunction with the causal interactions with its *immediate* environment wholly determine which computations the system performs. In other words, even if the intrinsic mechanism of a system underdetermines the set of automata that corresponds to the system's computations, taking into account the interaction between the system and its immediate context does the trick.

An example of the milder view has been defended by Piccinini who has pointed out that the individuation of computations may not only involve the intrinsic properties of mechanisms of a system, but also the interaction between mechanisms and their contexts. (cf. Piccinini 2008, 220). In Piccinini's view, however, "the wideness of putative computational properties of nervous systems does not even reach into the organisms' environment; it only reaches sensory receptors and muscle fibers, for that is enough to determine whether a nervous system performs computations and which computations it performs" (221). More specifically:

> "[T]he functional properties that are relevant to computational individuation, even when they are wide, are not *very* wide. They have to do with the normal interaction

[5] See also (Milkowski 2013: 79). Note, however, that Dewhurst in a recent paper (2018b) as well as Schweizer (2016) accept the view that extremely many systems perform computations.

> between a computing mechanism and its immediate mechanistic context via its input and output transducers. In the case of artificial computing mechanisms, the relevant context is, at one end, the relation between the forces exerted on input devices (such as keyboards) and the signals relayed by input devices to the computing components, and at the other end, the relation between the computing components' outputs and the signals released by the output devices. Those relations, together with the internal relations between components and their activities, determine whether a computation is performed by a mechanism and which computation it is." (2015, 140)

In other words, computational individuation requires more than the relevant mechanisms intrinsic to a computational system. But these further aspects exclusively involve the immediate causal-mechanistic interaction between the system and its immediate context. In an earlier paper, Piccinini uses an analogy to illustrate the idea:

> [P]lants absorb and emit many types of electromagnetic radiations, most of which have little or no functional significance. But when radiation within certain frequencies hits certain specialized molecules, it helps produce photosynthesis - an event of great functional significance. Without knowing which external events cause certain internal events and which external effects those internal events have, it may be difficult or impossible to distinguish the functionally relevant properties of a mechanism from the irrelevant ones. (Piccinini 2008, 220)

The analogy suggests that, just as the plant's production of photosynthesis depends on the frequencies of light radiation impinging on its surface, the computation performed by computational system partially depends on the causal interaction of the system and its context, or simply the physical inputs and outputs the system receives and emits. Once the causal inputs and outputs are sufficiently specified, it is unambiguous which of its various automata implemented by the system intrinsically speaking correspond to the computations that it actually performs.

The problem that we see with this line of reasoning is that it does not seem to apply to a number of examples of actual computing systems. Consider again the aircraft marshaller from section 2, and assume that in the normal environment of an ordinary airport what is relevant is his computation of an AND-gate: Only when both hands are up, the pilots of commercial flights direct the aircraft ahead (cf. figure 2). When the marshaller raises one hand only, the pilots assume that he is simply stretching his arm, or that he is still in the preparation phase for lifting up both arms. In both cases, they leave the aircraft at its current position.

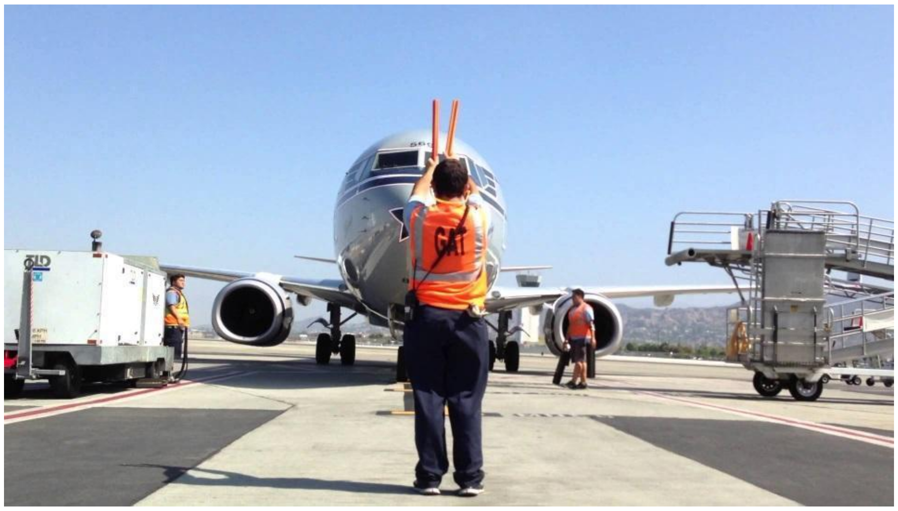

Figure 2: “Place yourself facing me!” in commercial airports

Moreover, assume that the same marshaller sometimes works on military aircraft carriers, which mainly carry helicopters (cf. figure 3). For the sake of the argument, assume that, fortunately, he does not have to learn any new rules for the processing of signals there. When he receives one beep on one side of his headphones, he lifts up one arm (not necessarily the same one). When he receives a beep on both ears, he lifts up both hands. In this context, however, one arm lifted up is relevant for the helicopter pilots as it means “wheel not properly out”, which immediately leads them to lift the helicopter up in the air again. The pilot performs the same move when both arms of the marshaller are up.

Figure 3: “wheel not properly out” on aircraft carriers

On the aircraft carrier, the air marshaller computes an OR-function. However, the interaction between the system (the marshaller) and its immediate context, i.e. the beep inputs and his hand-movement outputs, is exactly the same both in the aircraft carrier environment as well as the airport environment. Nevertheless, the marshaller computes an OR-function in the one, and an AND-function in the other context. It follows that the system plus its immediate causal environment are not generally sufficient for fixing the actual computations performed by the system.

A variant of the example of system P (cf. section 2) supports an argument in a similar vein. Consider a tri-stable volt gate that emits the outputs specified in table 5. In addition, assume that the outputs are functionally connected to a robot arm that moves up and down proportional to the volt inputs it receives. An output of 0-2.5V leads to an upward movement of the arm below 45 degrees, whilst an output of 2.5-5V leads to a movement between 45 degrees and 90 degrees. An output of 5-10 V leads to an arm movement above 90 degrees or the horizontal. In the first scenario (cf. scenario 1 in table 5), movements beyond the horizontal are interpreted as “great movement”, whilst any movement below 90 degrees is considered “No movement”.

Input 1	**Input 2**	**Output**	**Motor Output (scenario 1)**	**Motor Output (scenario 2)**

5-10 V	5-10 V	5-10 V	Great movement	Great movement
5-10 V	2.5-5 V	2.5-5 V	No movement	Little movement
5- 10 V	0-2.5 V	2.5-5 V	No movement	Little movement
2.5-5 V	5-10 V	2.5-5 V	No movement	Little movement
2.5-5 V	2.5-5 V	2.5-5 V	No movement	Little movement
2.5-5 V	0-2.5 V	2.5-5 V	No movement	Little movement
0-2.5 V	5-10 V	2.5-5 V	No movement	Little movement
0-2.5 V	2.5-5 V	2.5-5 V	No movement	Little movement
0-2.5 V	0 – 2.5 V	0-2.5 V	No movement	No movement

Table 5: Volt gate connected to robot arm movement.

Notice that with a more fine-grained specification of thresholds, the same gate computes inputs leading to three kinds of outputs (cf. scenario 2). Now movements between 0 and 45 degrees are considered as "no movement" as before, but movements between 45 and 90 degrees are considered to constitute "little movement". In both scenarios, the physical inputs are the same, namely different volt values. Moreover, in both outputs the physical outputs are the same since the arm moves identically in both scenarios. However, the computation is still not univocally fixed as in scenario 1 its syntactic output is 2-valued whilst in scenario 2 its syntactic output is 3-valued. This shows that the arm movements by themselves do not suffice to determine the units of the computation, and hence, the computation itself. And this shows that the normal interaction between a computing mechanism and its immediate mechanistic context via its input and output transducers together with the internal relations between components and their activities does not by itself "determine whether a computation is performed by a mechanism and which computation it is." (Piccinini 2015, 140). Or in short, specifying the internal mechanisms of a system – even those that are external to the computing components – is not sufficient to determine a system's computations.

What has just been established extends to real-life cases in science as well. Consider again Marr's (1982) celebrated example of a computational theory of vision. When Marr offered a computational model for how the retina solves the edge detection problem, he focused on those aspects of the retina that determine the sudden changes in light intensities in an input array in order to produce an edge image. The required computational function uses a Gaussian and a Laplacian operator ($[f(x,y) = \nabla^2 G * I(x,y)]$) in order to first smooth the image and to then compute the zero crossings of the resulting operation. That is, it calculates which points in the array are characterized by strong light value differences in their immediate environment.

Suppose now that the retina is transplanted into a larger device that performs heat detection such that the retina is used for a transmission of heat quanta, perhaps in the same way a water tube is used to transmit water quanta from the mountain to the valley. The reason why the retina can in principle be used for this task is that any bundle of light rays impinging on a retinal receptor will slightly alter the temperature of this receptor in the moment of its absorption. Each bundle of light rays will therefore transport a certain heat quantum into the retinal receptors. By physical preservation laws, this heat quantum will have to be transmitted somewhere in the eye, typically by the liquids floating through the retina. In this sense, the retina constantly emits certain heat quanta depending on having received as physical inputs certain light quanta. But, just as the water tube, in the new environment of the heat detection device the retina fulfills a function without performing any sensible computation. The mere transmission of heat quanta from A to B is no more a computation than the mere transmission of water quanta from A to B.

The transplanted retina may even receive exactly the same physical inputs as in the human eye (again the exact same array with the same values at each of the array points), and it may emit exactly the same physical outputs (electric signals and heat quanta). However, nothing about this physical equivalence suggests that the retina in the heat detection device computes anything at all. It may serve a biological or technological function in this new environment, perhaps in a similar way as the human heart performs a vital function. However, few cognitive scientists would be ready to consider the heart a computing system that processes bits of information only because it receives physical inputs and emits physical outputs. As before, the implication is that the computation performed by the retina is not fully accounted for by its mechanistic structure and its immediate causal context.

One may object that the retina performs the same computations, only the representations that the computation acts upon have changed. But again, it is not obvious that a retina transferred into a very different system performs anything like a computation. It is not obvious at all that the heat quanta inserted into the retina are representations in more than the trivial sense of being caused by, and causing, something outside of the retina. Hence, the computational identity of the retina is affected even though its local physical mechanism is not.

The conclusion suggested by these examples is that the interaction between a computational system and its direct causal context does not fully determine a system’s computations, contrary to the milder intrinsic mechanistic view. The switch that the marshaller underwent when he moved from one environment to the other, and that the retina experienced after having been moved into a heat detection device, shows that specifying the system intrinsically along with its immediate causal interactions does not always fully determine the computation that the system performs.

4 The Inessentialness of Mechanism

Let’s take stock. We have so far argued (section 2) for claim (i), namely, that contextual factors affect the computational identity of a computing system and that certain environmental aspects are required for computations accordingly. In section 3 we argued against a modified version of the “intrinsic” view according to which the missing external factors are causal interactions with the *immediate* environment. We showed that at least sometimes (some) relevant factors that contribute to computational identity are in the outside, non-immediate, environment of the system.

We now turn to our claim (ii), arguing that it is not necessary to specify the causal-mechanistic interaction between the system and its context in order to uncover a complete and adequate computational explanation. More accurately, our claim is this: Although some contextual factors contribute to the computational identity of the system (which is our claim (i)), computational explanations need not specify the causal-mechanistic interaction between the system and these contextual factors in order to offer a complete and adequate computational explanation.

We see at least three views one could hold on what makes a computational explanation of a system adequate vis-á-vis the system's context. First, one could hold the view that computational explanations do not have to take into account contextual factors at all, but that a specification of the intrinsic mechanistic structure of a system - possibly including its immediate inputs and outputs - is sufficient to adequately explain what and how the system computes. A second view might be that one has to specify the complete mechanism that relates the contextual factors with the computing system. This view seems to accord with Kaplan's dictum according to which "computational models possess explanatory force to the extent that they describe the mechanisms responsible for producing a given phenomenon." (2011, 339). A third view might specify that one has to take into account some "long-arm" or general aspects of the environment, but not the whole contextual mechanism. It is the last view, which can be described as the "NCC" account (for "non-causal contextual"), that we aim to defend in this section. By "non-causal" we do not mean that computational explanations have no causal components. Rather, our claim is that these explanations need not specify the causal mechanism that relates the system with its environment.

The NCC account actually consists of two claims, one positive and one negative. The positive claim is that satisfactory computational explanations necessarily take into account contextual features. As mentioned above, we will not defend this claim here, though we think that it follows naturally from claim (i) and some plausible adequacy conditions for explanations: If computational individuation partially depends on contextual features, it is reasonable that computational explanations refers to these contextual features as well.

The second, negative, contention is what in the introduction we have already called claim (ii): Satisfactory computational explanations need not specify the causal mechanism that connect the relevant contextual factors and the computing system. We do not claim that computational explanations never mention the whole contextual mechanisms, but merely insist that at least in some cases it is not necessary to do so for a complete and adequate computational explanation. We are not sure that someone has actually ever claimed to the contrary. The point seems to have been widely ignored in the philosophical literature. A fortiori we think that it is interesting and important to make our claim explicit. In the next section we will discuss the implication of this claim to the mechanistic view of computation.

To see why claim (ii) might be true, let us return to the example of the aircraft marshaller. A challenger of claim (ii) would argue that, to explain the computation performed by the marshaller, it is necessary to specify the marshaller's internal mechanisms along with the causal connections he bears to the helicopters that he directs, or to the aircraft carrier, perhaps even his causal connections to the whole military system that employs him. Once, the whole causal-mechanistic nexus surrounding the marshaller is taken into account, it is unambiguous which computations he performs. And, hence, it is necessary to specify the mechanistic interactions of the marshaller with his broader context to determine his internal computations.

But now let us modify the example of the aircraft marshaller a bit more, and let us assume that he sometimes receives and transmits his signals through entirely different mechanisms. For instance, assume that the marshaller in some of his work shifts sits in front of a screen in the airport tower and that he directs a robot down on the runway. The robot raises its robot arms if the marshaller presses the appropriate buttons. Moreover, the marshaller receives his inputs by two colleagues whispering "beep" into his ears.

It seems plausible that the marshaller performs the same computations as before. Yet, the mechanism that now connects him with the relevant contextual factors is a very different one. If the mechanism is different, and if computations are identical to their mechanisms, then, according to what above we described as the "second view" on computational explanations, the marshaller should be performing different computations in the two cases. The tension between these assertions shows that there is something problematic about the claim that it may be necessary to consider the complete interaction between mechanisms and their contexts. The contextual level has to be considered part of the computational level, but it is not clear that specifying the interactions between the mechanisms and the relevant contextual factors is the right way to spell out this inclusion.

A very similar point applies to the real-life cases of retinal vision (cf. section 3). For instance, the retina detects the edges of a butterfly hovering before the eye by determining the points that are characterized by sudden light changes in their environment. But now imagine a case, in which the environment-to-input mechanism is modified. In such a case, the retina may receive the same image from the same objects, but through a different mechanism. For instance, a camera might be catching a butterfly's image and upload it into the cloud. A tablet computer may then be placed in front of the eye with the downloaded image of the butterfly on its screen.

We would still say that in this case the retina computes edges of the actual butterfly's wings in front of it just as it did before when it was not mediated by a camera and a tablet computer. In other words, the retina realizes the same computations as before. However, the precise structure of the input mechanism was now very different, though this difference in mechanisms makes no difference to the computational explanation (which is the same).

We believe that this case is representative for many computational theories in cognitive neuroscience. Computational theories developed in this field seldom, if ever at all, specify the environment-system causal-mechanistic interaction. As an example, let us describe the edge detection case in a bit more detail.

In their computational theory of edge detection Marr and Hildreth (1980; Marr 1982) argue that the early visual processes compute the zero-crossing of second-derivative (Laplacian) operators (on the retinal images) in order to detect edges in the visual field, e.g., object boundaries. A central part of the theory is to explain why the visual edges that are the outputs of the computation are related to properties (e.g., object boundaries) in the visual field. At this point Marr and Hildreth appeal to *physical constraints*, which are contextual features in the physical environment of the perceiving individual (pp. 22–3). These features are facts about the physical environment we happen to live in. One such contextual fact is that intensity changes result "from surface discontinuities or from reflectance or illumination boundaries" (Marr and Hildreth 1980, 187). This fact serves to explain the appropriateness of computing zero-crossings of second derivative operators, as this computation highlights sharp changes in the retinal images ("visual edges"). Thus, by computing the zero-crossings of second derivative operators over the retinal images, the visual system captures the physical edges in the visual field. Nevertheless, the explanation itself makes no mentioning of the mechanism that relates the visual field with the visual system. Obviously, it is assumed that such a mechanism exists, but its specification is not part of the explanation. We would have exactly the same explanation as long as the zero-crossings of second derivatives reflect surface discontinuities (etc.), even if the environment-system mechanism were different.

One may ask what does the explanatory work here? Different people have different views about this (see, e.g., Rusanen and Lapi 2016; Egan 2017). Our own view, which we can only sketch briefly here, is that the explanation is grounded in mathematical similarities (some morphisms) between the visual system and the environment, in this case the visual field (see Shagrir 2010; Bechtel and Shagrir 2015; Shagrir and Bechtel 2017). On the one hand, the early visual processes compute some form of derivation (i.e., the zero-crossings of second-derivatives). On the other hand, the relevant relation in the world (visual field) is that of sharp changes in light intensities that typically occur along object boundaries. This relation can also be expressed, mathematically, by some form of derivation. Sharp changes of light intensities in the visual field are the extreme points of first-derivative or zero-crossings of second-derivative of the reflection function. We thus have derivation relations both in the visual system and the visual field, and this mathematical similarity explains the appropriateness of computing zero-crossings of second derivative operators. It is important to compute derivation because that this is the relation that also holds between magnitudes existing in the world, namely, sharp changes in reflection. Thus computing derivation is essential to identify edges. Computing factorization, multiplication, exponentiation and many other functions are not appropriate to the task because they they do not preserve the reflectance relation along physical edges, e.g., object boundaries. Thus computing these functions does not lead to edge detection. So, in short, the explanatory work in the case of edge detection is rooted in the similarities of certain mathematical properties of the visual system and the environment.

To conclude, the thought experiments of the marshaller and the butterfly as well as the reconstruction of the computational theory of edge detection presented in this section question the claim that the specification of environment-system causal mechanism is necessary for computational explanations. The causal-mechanistic interaction between the system and its context is not necessary to determine the computations performed by a computing system.

5 Contextual Computations and the Mechanistic View

How does NCC – the view that computational explanations necessarily take into account some "long-arm" or general aspects of the environment, but possibly not the whole contextual mechanism – affect the mechanistic view of computation? The answer largely depends on how we characterize the mechanistic view in general and the mechanistic view of computation in particular. The mechanistic view of computation is usually understood as asserting that computational explanations (models) are entirely mechanistic. Admittedly, it is not easy to say what "entirely" means here. But the general idea seems to be that (1) when the explanation specifies the mechanism, then the explanation is full-fledged. Nothing else is missing. Let us call this the "CMECH" view.

The CMECH view is seemingly held by Kaplan when he says that "computational models possess explanatory force to the extent that they describe the mechanisms responsible for producing a given phenomenon." (2011, 339) Piccinini has expressed views that converge to version CMECH of the mechanistic view as well. As he says on the locality or intrinsicness of computations: "In my view, computational explanation is a specific kind of mechanistic explanation—roughly, a mechanistic explanation that characterizes the inputs, outputs, and sometimes internal states of a mechanism as strings of symbols, and provides a rule, defined over the inputs (and possibly the internal states), for generating the outputs" (2006, 350). But, as mentioned above, we do not argue that Kaplan and Piccinini explicitly embrace CMECH.

It should be noted that CMECH, strong as it is, is weaker than the claim that: (2) All scientific explanations are (entirely) mechanistic. Let us call this the "AMECH" view. AMECH clearly implies CMECH ("AMECH -> CMECH"). However, the implication is not bidirectional. Someone can deny AMECH – believing that some scientific explanations are not (entirely) mechanistic – but still hold onto CMECH – the claim that computational explanations are mechanistic. Put differently, one can hold that computational explanations are entirely mechanistic (CMECH), but still deny AMECH by maintaining that some non-computational explanations are not entirely mechanistic. Claim AMECH has sometimes been attributed to

Craver and Kaplan (2011). Both authors now deny that they hold this claim, partially because they believe that etiological explanations are explanations as well (cf. Craver and Kaplan 2018, sec. 4).

How does NCC relate to CMECH and AMECH? The answer is that NCC implies the negation of CMECH. If computational explanations necessarily refer to contextual factors (claim (i) & some adequacy assumptions about explanations and metaphysics), and if the specification of the contextual dimension of computational explanations is not necessarily mechanistic (claim (ii)), then some computational explanations are not entirely mechanistic (not-CMECH).

This also means that NCC implies the negation of AMECH, since AMECH implies CMECH. Note, however, that there are two ways to deny NCC: One could choose to deny the positive claim of NCC (that computational explanations take into account contextual factors) in order to maintain CMECH; this seems to be Dewhurst's (2018a) strategy. But one could also deny the negative claim of NCC, namely our claim (ii) that computational explanations need not specify the causal mechanism that connect the relevant contextual factors and the computing system. Both strategies leave CMECH intact.

The NCC view, however, is consistent with less strict versions of the mechanistic view of computation. Among these is the view that (3) computational explanations are mechanistic but (at least some of them) have non-mechanistic explanatory aspects. The NCC view is also consistent with the even weaker claims that (4) some scientific explanations are not mechanistic, that (5) some computational explanations are not mechanistic, and that (6) all computational explanations are not mechanistic.

Our tendency is to believe in a version of view (3) based on the externalist conclusions established in sections 2 and 3. However, due limits of space we are not able to offer a comprehensive defense of such alternative view of computation (but see again our brief sketch of a positive theory of computational explanation at the end of section 4).

One could ask what differences the NCC view makes to explanatory practices, and whether these implications would be considered inappropriate by other mechanists. If is true that, whilst contextual factors play a role for computations and computational explanations, specifying the intrinsic mechanism of a system along with its causal interaction with its contexts is not necessary to explain computations (claim (ii)), then a general recommendation for explanatory practice in computational neuroscience is immediately established: Good and explanatory theories may be supplemented by details about the causal context of the investigated system. However, they need not be supplemented in this way, and it sufficient to explicate on the computational level of the system in question. In fact, adding the causal-mechanistic story, as some mechanists would suggest, will often amount to a simple proof of concept. In our view, this insight is widely embraced by computational neuroscience already. It is mainly in parts of the philosophical literature where an inaccurate picture of the requirements of computational neuroscience has established itself.

6 Conclusion

In this paper, our aim was to defend the beginning of an externalist view of computation, i.e. the claim that the contextual factors are indeed part of the computational level of computing systems, but that it is not necessary to specify the causal-mechanistic interaction between the system and its context in order to offer a complete and adequate computational explanation. We started by highlighting the mechanistic view of computation, which contends that computational explanations are mechanistic explanations. We noted, however, that mechanists disagree about the precise role that the environment plays for computational (mechanistic) explanations.

We focused on two main claims. The first claim contends that (i) contextual factors affect the computational identity of a computing system. The second claim (ii) says that, whilst contextual factors play a role for computations and computational explanations, specifying the intrinsic mechanism of a system along with its causal interaction with its contexts is not necessary to explain computations.

We argued for both claims (i) and (ii) on the basis of several thought experiments and Marr's celebrated theory of edge detection in humans. In a final step, we discussed the implications of our non-causal contextual (NCC) view for the mechanistic view of computational explanation. We showed that some versions of the mechanistic view of computation are consistent with our non-causal contextual view, whilst others are not.

Due to limits of space some important questions could not be answered in the present paper. We indicated that it is our view that computational explanations are mechanistic but (at least some of them) have non-mechanistic explanatory aspects. However, a thorough defense of this position will have to be left for future research. Moreover, our paper was mainly based on thought experiments. In a separate future study, we hope to offer independent support for our view using some more specific case studies from computational neuroscience.

References

- Bechtel, W. (2009). Looking down, around, and up: Mechanistic explanation in psychology. *Philosophical Psychology*, *22*(5), 543-564.
- 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*, 312–322.
- Block, N. (1990). Can the mind change the world? In G. Boolos (ed.) *Meaning and method: Essays in honor of Hilary Putnam*. Cambridge: Cambridge University Press, p. 137-170.
- Chalmers, DJ. (1994). On implementing a computation. *Minds and Machines* 4.4 : 391-402.
- Coelho Mollo, D. (2018). Functional individuation, mechanistic implementation: the proper way of seeing the mechanistic view of concrete computation, *Synthese*, 195, 3477–3497.
- Coelho Mollo, D. (forthcoming). Are there teleological functions to compute? *Philosophy of Science*.
- Craver, C., & Kaplan, D. M. (2018). Are more details better? On the norms of completeness for mechanistic explanations. *The British Journal for the Philosophy of Science*.
- Dewhurst, J. (2018a). Individuation without representation. *The British Journal for the Philosophy of Science* 69(1), 103–116.
- Dewhurst, J. (2018b). Computing mechanisms without proper functions. *Minds and Machines*, *28*(3), 569-588.
- Egan, F. (2017). Function-theoretic explanation and neural mechanisms. In D. M. Kaplan (Ed.), *Explanation and Integration in Mind and Brain Science* (pp. 145–163). Oxford University Press.
- Hopcroft, J.E., R. Motwani, J.D. Ullmann (2000). *Introduction to Automata Theory, Languages, and Computation*. Pearson Education.
- Kaplan, D., and C. Craver (2011). The explanatory force of dynamical and mathematical models in neuroscience: A mechanistic perspective. *Philosophy of scien*ce 78, 601-627.

- Marr, D. (1982). *Vision. A Computational Investigation into the Human Representation and Processing of Visual Information*. Freeman Press.
- Miłkowski, M. (2013). *Explaining the Computational Mind*. MIT Press.
- Miłkowski, M., (2017). The false dichotomy between causal realization and semantic computation. *Hybris. Internetowy Magazyn Filozoficzny,* 38, 1-21.
- Piccinini, G. (2006). Computational explanation in neuroscience. *Synthese*, *153*(3), 343-353.
- Piccinini, G. (2008). Computation without representation. *Philosophical studies*, *137*(2), 205-241.
- Piccinini, G. (2015). *Physical Computation: A Mechanistic Account*. Oxford University Press.
- Piccinini, G. (2017). Computation in Physical Systems, in Edward N. Zalta (editor). The Stanford Encyclopedia of Philosophy; Url: https://plato.stanford.edu/archives/sum2017/entries/computation-physicalsystems/
- Rusanen, A., & Lappi, O. (2016). On computational explanations. *Synthese*, *193*, 3931–3949.
- Schweizer, P. (2016). In what sense does the brain compute? In V. C. Müller (Ed.), Computing and philosophy. Heidelberg: Springer (Synthese Library).
- Shagrir, O. (2001). Content, computation and externalism. *Mind*, 110(438), 369-400.
- Shagrir, O. (2010). Marr on computational-level theories. *Philosophy of Science, 77*, 477-500.
- Shagrir, O. (2019). In defense of the semantic view of computation. *Synthese*, forthcoming.
- 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.
- Sprevak, M. (2010). Computation, individuation, and the received view on representation. *Studies in History and Philosophy of Science Part A*, *41*(3), 260-270.
- Sprevak, M. (2018). Triviality arguments about computational implementation. *Routledge Handbook of the Computational Mind* (edited by M. Sprevak & M. Colombo), Routledge: London, pp. 175–191.