In defense of the semantic view of computation

Oron Shagrir

Abstract: The semantic view of computation is the claim that semantic properties play an essential role in the individuation of physical computing systems such as laptops and brains. The main argument for the semantic view ("the master argument") rests on the fact that some physical systems simultaneously implement different automata at the same time, in the same space, and even in the very same physical properties ("simultaneous implementation"). Recently, several authors have challenged this argument (Piccinini 2008, 2015; Coelho Mollo 2018; Dewhurst 2018). They accept the premise of simultaneous implementation but reject the semantic conclusion. In this paper, I aim to explicate the semantic view and to address these objections. I first characterize the semantic view and distinguish it from other, closely related views. Then, I contend that the master argument for the semantic view survives the counter-arguments against it. One counter-argument is that computational individuation is not forced to choose between the implemented automata but rather always picks out a more basic computational structure. My response is that this move might undermine the notion of computational equivalence. Another counter-argument is that while computational individuation is forced to rely on extrinsic features, these features need not be semantic. My reply is that the semantic view better accounts for these extrinsic features than the proposed non-semantic alternatives.

1. Introduction

From laptops to smartphones, computing systems are everywhere today. Even the brain itself is considered, in brain and cognitive science, as a sort of computing system. Yet the nature of physical computing systems remains an open question. One finds in the literature very different accounts of physical computation.[1] In this article, I aim to defend the semantic view, which asserts that semantic properties are essentially involved with the nature of physical computing systems. A complete defense would consist of an explication of the view, a reply to the arguments against it and advancing arguments for it. My focus here is on the first and the third of these aspects.[2] Thus, I set out the semantic view and distinguish it from other, closely

[1] We can find the syntactic (Stich 1983), algorithmic (Copeland 1996), causal (Chalmers 2011) and mechanistic (Milkowski 2013; Piccinini 2015) approaches, to name just a few views of computation.

[2] Against the semantic view, one can argue that there are cases of computation without representation, that the semantic view endangers the objectivity of physical computation and that it jeopardizes the project of naturalizing

related, approaches (section 2), after which I turn to defend the main argument for the view ("the master argument"). This argument rests on the phenomenon of simultaneous implementation of different automata. Roughly speaking, the idea is that computational taxonomies must appeal to semantic properties in order to determine which automaton is relevant for the computational identity of the implementing physical system. Shagrir (2001), Sprevak (2010) and Rescorla (2013) have advanced versions of this argument. In section 3, I present a streamlined version of the master argument. Recently, Piccinini (2008; 2015), Coelho Mollo (2018), Dewhurst (2018) and others have challenged this argument. I discuss these objections, suggesting that the master argument prevails (sections 4 and 5).

2. What is a semantic view of computation?

A semantic view of computation states that semantic properties are somehow *essential* to the nature of computation.[3] Scholars use the terms "involve", "bear upon", "inform about", "are relevant to" and "have" (semantic properties) to capture the tight linkage between semantic properties and computation.[4] But what, precisely, do they mean? My aim in this section is to clarify some aspects of the semantic view. I will say something about essential involvement (section 2.1) and about semantics (section 2.2), after which I will distinguish the semantic view from other, closely related, approaches (section 2.3).

2.1 Essential involvement

There is a broad consensus that computation often operates on entities that carry informational or representational content. Yet this fact is perfectly consistent with non-

mentality. The first argument is advanced by Rescorla (2013; 2014) while Piccinini (2015: 34) makes the second argument, as well as the third (Piccinini 2008: 222). Other arguments against the semantic view have been proposed by Egan (1991; 1995). But see also Sprevak (2010), who challenges some objections to the semantic view.

[3] Thus, Fodor famously quipped that there is "no computation without representation" (1975: 37; 1980:122; Pylyshyn 1984: 62) and "no representations, no computations" (1975: 31); see also Dietrich (1990), Crane (1990), Churchland and Sejnowski (1992), Shagrir (2006), Ladyman (2009) and Sprevak (2010).

[4] Sprevak identifies the semantic view with the claim that "computation essentially involves representational content" (2010). Rescorla writes that "on the semantic view, *all* physical computational systems have semantic or representational properties" (2014). Piccinini says: "I call any view that computational states are representations that have their content essentially a *semantic account of computation*" (2015: 27).

semantic views.[5] The debate between the semantic and the non-semantic views is on whether these semantic properties play a role in the *individuation* of computation. Thus, both proponents and critics of the semantic view understand the locutions "involve", “bear upon” (etc.) in terms of the individuation, taxonomy or identity conditions of computation, where the individuated entities can be systems (Piccinini, Rescorla), processes (Sprevak, Dewhurst), states (Piccinini, Dewhurst) or events. The semantic view claims that the individuation of these systems (etc.) takes into account their semantic properties. This means that semantic properties play a role in determining whether a certain system (etc.) is computing or not, whether two systems (etc.) are computationally similar or computationally different, whether or not changes in semantic properties alter computational identity, and so on.[6]

I said that the term "involve" (etc.) is to be understood in terms of individuating computation. We next ask what is meant by *essential* in the phrase "essentially involve". The simple answer is that *essential* means *always*, where "always" refers to any computation, whether actual or possible.[7] So, the semantic view asserts that semantic properties *always* impact the individuation of computation. The demand is not that the individuation take into account all semantic properties or only semantic properties (Piccinini 2015: 27). The claim, rather, is that some semantic properties, perhaps with other non-semantic properties, always affect (or "impact", or "play a role in" or "enter into") the individuation of computation.

---

[5] Thus, Frances Egan, who holds a non-semantic view writes that "computational theories treat human cognitive processes as a species of information processing" (1995: 181). Piccinini, who argues against the semantic view, writes: "In our everyday life, we usually employ computations to process meaningful symbols, to extract useful information from them" (2015: 26).

[6] Curiously, some scholars hold that semantic properties play an individuative role in distinguishing between computing systems, such as brains and desktops, and non-computing systems, such as stomachs and washing machines. But they do not think that semantic properties play an individuative role in distinguishing between different kinds of computing systems, like brains and desktops. In other words, in their view, semantic properties play no role in determining computational equivalence, namely, the classification of *computing* systems, processes, events and states into computational types or kinds. Fodor, for example, thinks that a representation is a criterion distinguishing between computing and non-computing systems (1975: 75; Crane 2015: 154). Clearly, however, he thinks that computational *kinds* are distinguished according to their syntactic properties, whereas syntactic individuation does not require semantic individuation (1994: 8). I leave aside the question of whether this view is a consistent one. My aim here is to defend the seemingly stronger view that semantic properties affect computational equivalence.

[7] We can ask about the scope of *possibility*, namely, whether "always" refers to any physically, metaphysically, or even logically or conceptually *possible* computation. Given that our focus is physical systems, we can be satisfied with *physically possible* computations.

The semantic view asserts that semantic properties *always* affect computational individuation. A *non-semantic view* asserts that semantic properties *never* affect computational individuation. As mentioned above, a non-semantic view is consistent with the claim that computation often *operates* with semantic properties. What it rejects, however, is that computational *individuation* ever takes into account semantic properties.[8] Some philosophers argue that computational individuation takes into account semantic properties in some cases but not in others. We can call this view *neither semantic nor non-semantic* (NSNNS). Rescorla (2013) explicitly argues for NSNNS. Burge seems to uphold this view as well. On the one hand, Burge famously argues that visual content affects the individuation of computational-cognitive states (Burge 1986; 2010). On the other, he also says, when discussing a specific computation, that "here we have computation *without* representation" (Burge 2010: 424). So, both Rescorla and Burge seem to think that we can have computation without representation; in this respect, their view is not semantic. But they also seem to think that when computation accompanies representation, the content of the representation at least sometimes impacts computational individuation; in this respect, their view is not non-semantic.[9]

2.2. Semantics

Unsurprisingly, the semantic view refers to semantic features, usually properties. But what is meant by *semantic*? This is a notoriously hard question when considering semantics in general. The short answer, however, is that, when confined to computing systems, semantic properties refer to representational or informational *content*. The semantic view claims that the individuation of computational systems (etc.) makes an essential reference to the content of the states of the system.

But what is this content? I think that most would agree that content involve *aboutness*. The states of the computing system that have content denote or refer to some other objects, events, properties, etc. The latter entities can be in the environment of the (computing) system,

[8] An important category of non-semantic views associates computational individuation with information, but whereas most people would characterize information in semantic terms, the proponents of these non-semantic views think that *information* is not a semantic kind (Milkowski 2013: 48; Fresco 2014).

[9] Lee (forthcoming) proposes a pluralistic view about computational identity. The view is similar to NSNNS in the claim that semantics affects computational individuation in some cases and does not in other cases.

but also in some distant, counterfactual or abstract domains, as well as within the system itself. Some identify content with the referents themselves, whereas others identify it with some perspectives (e.g., "senses") of these referents. All agree, however, that aboutness implies *directionality*: While objects with content refer to some other entities, the latter entities might not refer back, and they might not have semantic properties at all.

More controversially, we can ask what kinds of content play a role in computing systems, which factors fix content, and whether aboutness is sufficient to characterize content. Following others, I suggest taking a *pluralistic* stance. One aspect of this pluralism is that different computing systems might operate on different kinds of semantic properties (Piccinini and Scarantino and 2011; Piccinini 2015, chap. 14). Some computations operate on (so-called) representations whose content is interpretative, in the sense that it is derived by an (usually external) observer, designer or a user. Laptops operate on semantic properties that are defined, at least partly, by the user of the machine. Other computations might operate on representations whose content is non-derived. Presumably, the content of the computations that take place in our brain is not defined by the interpretation of an external observer. Another aspect of pluralism pertains to the factors that determine content. In classical cognitive science, the content of a computing system might receive a functional, model-based, treatment.[10] Others, specifically neural computation, might operate on representation or information whose content is, at least partly, causally based, or so it is often assumed in cognitive neuroscience.[11] It might turn out that there is a single account of the content of computing systems. The semantic view is perfectly consistent with this scenario. But it is also consistent with the far more reasonable scenario that computation allows for different kinds of semantic properties.

Sprevak (2010) suggests that the semantic properties involved with computing systems might not always have a particularly complex structure (*minimalism*). Some computations operate on

[10] See Cummins (1989) and Ramsey (2007); a comprehensive treatment of the functional view is provided, e.g., by Block (1987).

[11] David Marr (1982), for example, writes that “the apocryphal grandmother cell” (p. 15) is “a cell that fires only when one’s grandmother comes into view” (note on p. 15); a comprehensive treatment of the causal view (augmented with a teleological component) is provided, e.g., by Dretske (1988).

propositional and compositional representational systems but many others seem to operate on representational systems that lack these rich structures. Cells in V1, for instance, do not seem to have a complex propositional structure. Another facet of minimalism pertains to internal representations. Some computations involve internal representations, but others do not. Two-layer feed-forward networks map input representations to output representations, but the mapping involves no internal ("hidden-layer") representations. Finally, some computations, though operating on internal complex propositional structures, do not have mental content.

Some philosophers think that the semantic properties involved in computation have a *normative* aspect (Cummins 1989; Ramsey 2007). Normativity implies that the representation can be right or wrong, and that there is a possibility of *misrepresentation*. My representation of cow, *R*, might be tokened by a horse under some darkish conditions. This does not (necessarily) mean that *R* is a representation of cow-or-horse. *R* is still a representation of cow; in the described case, *R* misrepresents the horse as a cow.[12] The normativity of semantic properties is usually associated with some *function*, in the sense of goal or purpose (Millikan 1984; Dretske 1988). My view is that the representational (or informational) content in computing systems is *always normative*, in the sense that there is an issue of right or wrong in their application. But I will not argue for this claim here.

To sum up, by *semantic* we refer to the informational or representational *content* of computation (states, systems, processes, etc.), which means that these states come with directional reference (*aboutness*). We do not require that there be a single account of the content of all computing systems; there might well be different kinds of content as well as different determinants of content (*pluralism*). The pertinent representation or information might not be mental, propositional or compositional, nor even internal (*minimalism*). According to some philosophers, content always involves the possibility of misrepresentation (*normativity*).

[12] Dretske (1988) distinguishes between semantic properties that have a normative aspect ("representation") and semantic properties that do not ("information"). Hence, according to Dretske, there is no misinformation: *R* carries the information that there is a horse in front of me, even under the darkish conditions in which I misrepresent this horse as a cow. Other scholars do not accept this distinction, introducing also cases of misinformation (Piccinini and Scarantino 2011).

2.3 What the semantic view is not

Let us distinguish the semantic view from closely related approaches. First, the semantic view of computation is consistent with a non-semantic view about implementation. A non-semantic view about implementation asserts that the relation of implementing an automaton by a physical system does not involve semantic properties (e.g., Chalmers 2011). A semantic view of computation is consistent with this assertion. Of course, if you think that computation *is* (nothing but) the implementation of an automaton, then you cannot hold a semantic view about computation *and* a non-semantic view about implementation. But the semantic view of computation is not committed to the identification of computation with implementation. A proponent of the semantic view of computation is free to hold that the implementing of an automaton (or some formal structure more generally) is necessary for computation and yet is non-semantic. She can say that counting the implemented automaton as a computational structure – and counting the implementing physical system as a computing system – essentially involves semantic properties.[13]

The semantic view can also be distinguished from the view that *computational descriptions* (e.g., in theories and explanations) make specific reference to semantic properties. The proponents of the semantic view can surely hold that there are non-semantic descriptions of computing systems. They are also free to hold that computational descriptions themselves are formulated in formal, e.g., mathematical, terms, and do not make specific reference to semantic properties. The semantic view is the claim that counting the described system as computational essentially involves semantic properties.

The semantic view of computation is distinct from the view that computing processes are not sensitive to semantic properties.[14] Many scholars hold that the computing processes in the laptop operate on symbols or bits, and yet these processes are completely "blind" (as opposed

[13] Note that I do not *argue* that implementation must be non-semantic. If implementation is semantic, then (under reasonable assumptions) computation is semantic too. My claim is that one can hold a semantic view of computation without upholding a semantic view of implementation. My argument, in other words, is that even if automata and their implementations can be individuated non-semantically (as I think is true), computation is still semantic.

[14] This distinction is made by Piccinini (2008), Sprevak (2010) and Rescorla (2012).

to "sensitive") to the content of the symbols. In this sense, computing processes are not sensitive to semantic properties. Some, assuming that the claims about individuation and sensitivity are closely related, present this blindness as a refutation of the semantic view. But I think that both proponents and opponents of the semantic view nowadays agree that the two claims are separate. The semantic view is a claim about the sort of properties that matters to the individuation of computation and not about the properties to which computation is sensitive.

The semantic view differs from externalism about computation, namely, the claim that the individuation of computation essentially takes into account features that are external – located outside – the computing system (etc.). Both of these approaches are concerned with individuation. However, externalism is not committed to the claim that the external features are semantic, whereas the semantic view is neutral about the so-called semantic internalism/ externalism debate.[15]

Last, but not least, the semantic view should be distinguished from naturalistic theories of content. The semantic view, though consistent with a naturalistic approach to content, is not committed to naturalism about mental content, and it is certainly not committed to naturalism about semantic properties of computing systems. The semantic view is a claim about the identity conditions of computation: It asserts that content essentially determines the identity of a computing system (whereas a view that is not semantic denies that content has this individuative role). Whether content itself can be naturalized or not is an important but separate issue. To compare: Take the debate on whether computation is sensitive to content or not. As far as I can tell, this issue is orthogonal to the debate between naturalists and their opponents. Computation would (or not) be sensitive to content regardless of whether content

[15] Authors who argue for externalism, without committing (or even objecting) to the semantic view, include Bontly (1998), Horowitz (2007), Piccinini (2008; 2015), and Shea (2013). We can also distinguish externalism about computation from computational externalism (I am indebted to a referee about this point). Computational (or wide) externalism is a claim about the location of the vehicles of computation (e.g., Wilson (1994), whereas externalism about computation is a claim about what individuates computational states, regardless of where they are located.

can be naturalized. The same goes for computational individuation. Content would affect (or not) computational individuation regardless of whether content can be naturalized.[16]

3. The master argument for the semantic view

There are several arguments for the semantic view.[17] My focus here is what I call the *master argument*. Versions of the argument were introduced by Shagrir (2001), Sprevak (2010) and Rescorla (2013). Here, I will present a streamlined variant of my version of the argument. It goes like this:

(1) A physical system might simultaneously implement several different automata S_1, S_2, S_3,….

(2) The contents of the system's states determine (at least partly) which of the implemented automata, S_i, is relevant for computational individuation.

Conclusion: The computational individuation of a physical system is essentially affected by content.

Some comments: First, I assume here that implementing a formal structure is necessary for computing. The implemented formal structures can be algorithms, graphs or even formal dynamics. My focus here is on automata. Second, I do *not* assume that implementing a formal structure (e.g., an automaton) is a semantic relation; in fact, I take it that the implementation is a non-semantic relation between a physical system and an automaton. My claim is that turning an automaton into a *computational structure* involves semantics: A computational structure is just the implemented automaton that is taken into account by computational individuation. Last, the second premise says that a computational taxonomy might not take into account all the implemented automata. Typically, the taxonomy would take into account only one

[16] It is nevertheless true that if one is a naturalist about computation (or even just mental computation), then her motivation to adopt the semantic view will depend on whether or not we succeed at naturalizing (mental) content.

[17] A main motivation for the semantic view is that it arguably provides an easy solution to the problem of distinguishing computing from non-computing physical systems; see Sprevak (2010) for a discussion of this and other arguments for the semantic view.

automaton (in a given context). But there might be cases in which the taxonomy takes into account more than one.

The argument can be seen as a claim about *computational vehicles* (Rescorla 2015; Shea forthcoming). From this perspective, the assertion is that the identity conditions of computational vehicles essentially involve semantic properties. This does not mean that we cannot individuate vehicles in non-semantic terms. A vehicle is (in the token-identity sense) an implemented formal structure (e.g., the automaton), and as such, can be individuated non-semantically. The claim is that when we treat this implemented automaton as a *computational vehicle*, semantics sneaks in. Semantics gets into the picture when we classify the implementing physical states into computational types.

3.1 Simultaneous implementation

Simultaneous implementation is the phenomenon in which the same token physical system implements different automata at the same time, in the same space and even with the very same physical properties. While others describe simultaneous implementation at different levels of organization (Chalmers 1996; Scheutz 1999; 2001), I offer examples of simultaneous implementation at the same level of organization. Many describe automata in terms of "total states" (e.g., Chalmers 1996). I describe them in terms of gates, which are the bases of real digital computing. The two descriptions are equivalent (Minsky 1967: 55-58).

Consider a physical system **P** that works as follows: **P** emits 5-10 volts if it receives voltages greater than 5 from each of the two input channels; 0-2.5 volts if it receives under 2.5 volts from each input channel; and 2.5-5 volts otherwise. I will use the symbols H, L, and M to signify these three different physical properties (5-10 volts; 0-2.5 volts; 2.5-5 volts), which implies that the argument does not depend on these specific physical properties. The behavior of **P** is summarized in Table 1. Now, grouping the inputs and outputs around the emission/reception of 0-5 volts (L-M) and of 5-10 volts (H), and assigning the symbol ‘0’ to the emission/reception of L-M and ‘1’ to the emission/reception of H, **P** is implementing the *AND-gate* (Table 2). But grouping the inputs and outputs around the emission/reception of L and of M-H, and assigning

the symbol '0' to the emission/reception of L and '1' to the emission/reception of M-H, **P** is implementing the *OR-gate* (Table 3).[18]

I do not claim that we can create a functionally complete set of dual gates. Indeed, I do not even know how to build dual gates that include *NAND* or *NOR*. Nevertheless, we could create other dual gates in a similar way – for example, one that simultaneously implements *OR* and *XOR* (Shagrir 2001). Another interesting possibility is of a physical system **Q** that simultaneously implements the *AND* gate twice (Table 4). Receiving M in both input channels, **Q** simultaneously performs the '0','0' $\rightarrow$ '0' mapping under the first implemented *AND-gate*, but the '1','1' $\rightarrow$ '1' mapping under the second implemented *AND-gate*. We can simultaneously implement more complex automata. Consider an automaton that consists of *AND* and *XOR* gates; such an automaton can be used to add 1-digit numbers (Block 1995). If we implement this automaton using dual *AND*/*OR* and *XOR*/*OR* gates, the physical system simultaneously implements an automaton that consists of two *OR* gates (Shagrir 2001). Alternatively, consider an implementation of a series of four *AND-gates* (we can assume that the inputs to the gates are the output from the preceding gate and an external channel). One way to implement this series is with a sequence of four physical systems, **P**, each of which implements the *AND-gate*. But **P** also implements, simultaneously, the *OR-gate*. Thus, the sequence of the four physical, **P**, systems, simultaneously implements a series of *OR-gates*. A more interesting implementation of the series of *AND-gates* is with the sequence of physical systems, **P**-**Q**-**P**-**Q**. This sequence simultaneously implements the alternating, *AND-OR-AND-OR*, operations.[19]

Let us compare simultaneous implementation with Putnam's and Searle's triviality results. Basically, Putnam (1988) and Searle (1992) hold that almost every physical system implements every automaton. These authors have been accused of assuming an overly liberal notion of implementation (e.g., Chalmers 1996). Some scholars have suggested that we can avoid the triviality results by adding causal, modal and grouping constraints to the abstract-to-physical

[18] Obviously, real flip-detectors need some voltage-range (say from 4 to 6 volts) to distinguish between the two classes of voltage-states. I ignore this for the sake of simplicity.

[19] One might think that the different Boolean functions combine at the end into logically equivalent functions. Fresco, Copeland and Wolf (forthcoming) prove, however, that the likelihood of this scenario is close to nil for a physical system, such as this one, which implements a function on five or more dual gates.

mapping relation.[20] I do not challenge this conclusion; in fact, I assume that we can reasonably account for implementation in non-semantic terms. Simultaneous implementation is the much weaker claim that some physical systems simultaneously implement more than one automaton. Simultaneous implementation does not imply that every physical system implements more than one automaton. Nor does it imply that rocks implement complex automata.

My construction differs from Putnam's and Searle's in that simultaneous implementation is achieved through the very same physical properties of **P**, namely, its voltages. Furthermore, in the Putnam and Searle cases, there is an arbitrary mapping from different physical states to the same states of the automaton. In our case, we keep consistent the mapping from the physical properties of **P** to the same inputs/outputs of the gate. The different implementations result from coupling different voltages (across implementations, not within implementations) to the same inputs/outputs. But within implementations, relative to the initial assignment, each logical gate reflects the causal structure of **P**. In this respect, we have a fairly standard implementation of logical gates that satisfies the conditions set by Chalmers and others on implementation. Indeed, the proposal seems to accord with the standard ways in which we implement bits of 0's and 1's in physical systems.

3.2 From simultaneous implementation to the semantic individuation of computational states

What follows from simultaneous implementation? Concerning the notion of implementation – not much. In particular, I do not think that simultaneous implementation gives us a reason to adopt a semantic conception of *implementation*. A theory of implementation (e.g., Chalmers 2011) can and does tolerate simultaneous implementation, or at least I do not assume otherwise. This seemingly innocuous claim, however, might have far-reaching consequences for theories of computation. Here, I use simultaneous implementation as a premise for an argument for the semantic individuation of computational states.

[20] The critics disagree about the constraints on implementation that should be added to the simple mapping account. These include causal (Chrisley 1994; Melnyk 1996; Chalmers 1996, 2011), modal (Chalmers 1996, 2011; Copeland 1996; Scheutz 1999, 2001), dispositional (Klein 2008), mechanistic (Piccinini 2007, 2015; Milkowski 2013) and pragmatic (Egan 2012; Matthews and Dresner 2016) constraints. Others put further restrictions on the ways we group physical objects and properties into types of states (Scheutz 1999, 2001, 2012; Godfrey-Smith 2009).

Consider our physical system **P**. We recall that when assigning '0' to the emission/reception of L-M and '1' to the emission/reception of H, **P** implements *AND*. However, when assigning the symbol '0' to the emission/reception of L and '1' to the emission/reception of M-H, **P** implements *OR*. Which automaton is relevant to computational individuation? I contend that if the content (e.g., the interpretation) of L and M is (the number) 0 and the content of H is 1, then **P** falls under the computational kind *AND*. If the content of L is (the number) 0 and of M and H is 1, however, then **P** falls under the computational kind *OR*. This indicates that content matters to the computational structure of **P**.

Or consider some sensor that simultaneously implements two automata, S_1 and S_2. Suppose that some input/output variables for S_1 are correlated with the physical properties L and M, and the other input/output variables for S_1 are correlated with the physical properties H. Also suppose that some input/output variables for S_2 are correlated with L and the other input/output variables for S_2 are correlated with M and H. I contend that S_1 will be preferred over S_2 if it turns out (say) that outputting L-M has one content (e.g., apples) and outputting H has another (e.g., oranges). I am thus suggesting that the contents correlated with the physical properties L, M and H, determine, at least partly, which automaton is taken for individuative purposes. A computational taxonomy will select the automaton, e.g., S_1, whose implementing physical states correlate with the contents of these states.

To see this point more vividly, assume now that our sensor, which simultaneously implements S_1 and S_2, is removed to and embedded in a very different environment. In this environment, the contents of the physical properties, L, M and H, fit not with S_1, but with S_2. In this environment, the contents of L and H remain the same, but the content of M is now the same as the content of H (e.g., oranges instead of apples). I think that, in this scenario, we would say that the automaton that is relevant for computational individuation is no longer S_1, but S_2. It is quite reasonable to say that in the new environment, a computational taxonomy will take into account S_2 and not S_1. In other words, it will take into account the automaton that fits with the current content. This example shows that a change in the content of M can alter the computational structure of the system from S_1 to S_2. Thus, content impacts on computational identity.

Thus far, I have associated computational identity with the content of input and output representations. Often, this will do. But there are cases in which we have no choice but to appeal to the content of internal states.[21] Here is an example: A physical system simultaneously implements two automata. But the inputs and outputs of both automata are correlated with the same input and output, L and H. In this case, the content of the input and output states cannot distinguish between the implemented automata. Instead, we have to look into the content of *internal states*. For example, we can assume that a certain physical state, **a**, implements a state, A, of one automaton, but two sub-states of **a**, $\mathbf{a_1}$ and $\mathbf{a_2}$, implement the states A1 and A2 of the other automaton. In this case, we would favor the first automaton over the second if only one content is correlated with **a.** We would favor the second automaton over the first if two different contents are correlated with **a**, one content with $\mathbf{a_1}$, and another with $\mathbf{a_2}$.

I do not claim that every change in content alters computational identity. Assume that, in our visual system, outputting L and outputting M have green content and outputting H has red content. Changing the content of *these* physical types (i.e., of L, M and H) to apple content and orange content will not alter the identity of the computing system. A computational taxonomy will still take into account the same S_1 automaton. Sometimes, however, changing the content results in computational change. Attributing apple content to outputting L and orange content to outputting M and outputting H will alter computational identity. A computational taxonomy will now take into account the S_2 automaton. The point, in a nutshell, is that the content of the physical states/properties (e.g. voltages) matters to how we group these states/properties together into computational types. More specifically, the sameness and difference of the contents of L, M and H (in a given context) is a crucial factor in grouping these physical properties into computational types. Altering the contents of L, M and H will result in a new grouping of computational types, in accordance with the new contents. In some cases, the new

[21] This point was raised by Jack Copeland, who refers to simultaneous implementation as "the indeterminacy of computation". Notably, there are other cases like these where the internal states have no semantic content at all. In these cases, I would say that the computational identity of the system is "truly" indeterminate.

grouping will be the same as the old one, and in some cases, it will be different. Either way, content drives, at least partly, the formation of computational types.

My view is situated somewhere between that of Burge (1986; 2010) and that of Egan (1995; 2010). Like Burge, I think that computational identity is content-sensitive and can vary across contexts. Unlike Burge, I do not assume that every change in content makes a computational difference. Like Egan, I think that changing the content of the states of the same computational vehicle does not affect computational identity. Unlike Egan, I do not take this to show that computational identity is content- (and context-) independent. A change in content can alter the computational vehicle, and with it, computational identity.

Let us turn to two objections to the argument. Both objections target the second premise, namely, the claim that content determines which automaton is relevant to computational individuation. According to the first objection, it is a mistake to assume that a computational taxonomy takes into account one of the implemented automata, or even automata at all. The computational structure of the system is identified with some more basic (non-semantic) structure. According to the second objection, it is agreed that a computational taxonomy takes into account one of the implemented automata and that this identification depends on extrinsic features; the objection is to the claim that these features are semantic.

4. Objection 1: Individuation goes more basic

Some scholars have argued that a physical system has a more basic (non-semantic) computational structure than those presented in the argument. According to this view, the mistake in the argument is in assuming that we have to choose between the implemented automata, e.g., between *AND* and *OR*. We don't need to make this choice: The computational structure of the system is identified by none of these functions/automata. This does not mean that other implemented automata and logical functions are uninteresting. They might be useful for all sorts of applications in computer science and engineering. But their individuation "is over and above computational individuation… Computational individuation is more basic, and non-semantic" (Coelho Mollo 2018: 3492).

What is this more-basic structure? We find several proposals in the literature. One is a *maximal* automaton. An automaton is maximal for a given system just in case it is not implied by another, more complex automaton that the system implements. For example, the maximal automaton implemented by the physical system **P** is the one associated with tri-stable gates (Table 1) and can be described by Table 5. Now, it is a mathematical fact about automata that when a system implements some (maximal) automaton it *ipso facto* simultaneously implements simpler automata. For example, by implementing the automaton with the tri-stable gates, **P** also simultaneously implements (under some re-labelling) the simpler automata *AND* and *OR*.[22]

Joe Dewhurst (2018) has advanced another proposal. Taking the mechanistic viewpoint, he argues that computational identity is defined not by the logical functions (*AND*, *OR*, etc.) but by the computing mechanism. Take our physical system **P**. According to Dewhurst, the computational identity of the system is provided by Table 1. This table provides a description of the computing mechanism, namely, the components of the system, their functions and interactions. This description tells us that the system consists of two processor types and three digit types, and it also tells us how these components interact with each other. This is everything we need to know for the purposes of computational individuation.

Dimitri Coelho Mollo (2018) has put forward a third suggestion. Like Dewhurst, he, too, adopts the mechanistic view of computation. But he correctly observes that Dewhurst's proposal, which ties computational identity with specific implementational (structural) physical properties, e.g., voltages, is untenable; the individuation conditions are too fine-grained. Consequently, systems whose implementational physical properties are different cannot be computationally equivalent. Coelho Mollo (2018) suggests identifying the computing mechanism with the functional profile of the system **P**. This profile types different implementational properties under the same equivalence classes and thus allows for some

[22] Chalmers (1996) and Scheutz (2001) introduce and discuss maximal automata in the context of simultaneous implementation.

multiple realization.[23] The functional profile of **P** is given by Table 5, where the labels A, B and C simply stand for three equivalent classes of implementational properties.

Last, one can argue that that the more "basic" structure is the set {S1, S2,…} of all implemented automata (this has been suggested by Milkowski 2013).

Reply: Let us first clarify what is at stake. Two suggestions have been put on the table in the face of simultaneous implementation. One is that computational individuation might take into account one of the implemented automata (and that this one is determined by content). The other is that computational individuation always takes into account some basic structure (which is non-semantic). This structure might be a maximal automaton, the mechanistic structure of the system, its functional profile, or the entire span of implemented automata. The debate is not so much about semantic vs. non-semantic individuation as it is about less basic (e.g., non-maximal automaton) vs. more basic (e.g., maximal automaton) individuation. Thus, Piccinini – whose views are discussed in section 5 – thinks that computational individuation might take into account a non-basic structure (e.g., *OR*) but that the individuation is still non-semantic. My reply is that the proposals grouped under the second, basic-structure suggestion all suffer from the same drawback: They do not do justice to, and even put at risk, the notion of computational equivalence, and, hence, of computational individuation.

Let us return to the sensor example, which simultaneously implements S_1 and S_2. Assume that you have another sensor that is somewhat physically different from mine. It implements S_1, but not S_2. We can assume that it implements S_1 in two different physical properties, L* and H*. Also assume that the contents of the two sensors align (as described above) with S_1. In my sensor, one kind of content (e.g., oranges) is aligned with M-H and the other kind of content with L (e.g., apples). In your sensor, one kind of content (e.g., oranges) is aligned with H* and another one with L* (e.g., apples). I think that there is little doubt that a computational taxonomy will count the two sensors as computationally equivalent. It will count them as equivalent because they both implement the same automaton, S_1, in its sensing task. Note that

[23] Like Dewhurst, Coelho Mollo thinks that the implementational details are part of computational explanations, but he distinguishes between functional and implementational levels.

the individuation is not merely in terms of representational commonalities and differences. As said above, computational individuation does not require that the two sensors have exactly the same contents. Assume that your sensor, while implementing S_1, has other, color-contents; we might even consider the semantic tasks as different. We would still say that the two sensors are computationally equivalent as they implement the same automaton, S_1, in their tasks.

Another way to make the point is this: Assume that we manufacture a set of sensors that fulfil their sensing goal by implementing S_1. I think that we would happily deem them as computationally equivalent. Discovering, one day, that the manufactured devices are somewhat different and that some of them simultaneously implement S_2, others S_3, and so on, would not change the equivalence verdict. Given that they all keep sensing by implementing S_1, we would still deem them as computationally equivalent. The fact that some sensors implement other automata would be ignored for the purposes of computational individuation. I think that this practice reflects the way we individuate computing systems in general. If I'm right about this, then computational individuations operate at the level of the implemented automata, at least in some cases, and not at a more basic level.

Thus far, I have argued that computational individuation takes into account one of the implemented automata rather than a more basic structure. I think, moreover, that the proposals grouped under the second (basic-structure) suggestion jeopardize the notion of computational equivalence, and hence, of computational individuation. They all imply that different physical systems might always turn out to be computationally distinct. All it takes is one cell (say) flipping at one more value. For example, if our flip-detector turns out to differentiate between 0-1 volt-inputs and 1-2.5 volt-inputs, then we are dealing with a system with a different "basic" computational structure, which is computationally different from the system we started with. And, of course, there is no reason to limit computation to digital cases. In analog cases, we can carve up the values in many more ways as we are no longer limited to the digital threshold values. If that is the case – and assuming that one takes seriously the notion of analog computation – we might find out that different individual systems always belong to different computational types.

Coelho Mollo (2018) and Dewhurst (2018) both discuss the possibility that different physical systems are inherently computationally distinct. Coelho Mollo denies that this possibility presents a problem for computational individuation:

> In consequence, devices that differ in the number of stable states (e.g. two vs. three), as in Shagrir's (2001) version of the argument from the multiplicity of computations, are never computationally equivalent… This, I take, is as it should be: given their different functional profiles, those two devices will differ in their capacity to carry out logical and mathematical functions—having a richer functional structure makes the tri-stable device considerably more versatile (2018: 3494-5, n. 20).

Importantly, Coelho Mollo (and Dewhurst too) thinks that the systems can be equivalent under some other, non-computational, schemes. In particular, they both agree that the two systems might share *semantically individuated* logical functions (and other automata). Coelho Mollo writes that the "individuation by logical function …may well rely on wide functions or semantic properties" (2018: 3492). Dewhurst says that "both Shagrir and Sprevak are correct when they point out that the logical status of a gate is indeterminate prior to the attribution or identification of its representational content" (2018: 107). But they both insist that this individuation scheme is not computational.

There is no debate here that systems that differ in the number of stable states (e.g. two vs. three) are not equivalent in some respects: They differ in the totality of automata they implement. I also agree that this difference is reflected "in their capacity to carry out logical and mathematical functions". In my view, however, this does not oblige us to say that the system must be computationally different. I shall explain my position while focusing this time on the notion of computational explanation. Take the tri-stable device that has the capacity to carry out *AND* and *OR*. Assume, as before, that the device performs some sensing task and that it exercises *AND* to perform this task. I think that it is reasonable to say that the fact that it exercises *AND* explains (at least partly) this sensing task. The explanation itself might be blind to the fact that the device is tri-stable; we would provide the same explanation for a bi-stable device that exercises *AND* to perform the sensing task.

Note that the explanation itself is a formal one and does not mention specific contents (e.g. apples and oranges); the explanation shows how performing the *AND* function supports the

sensing task. Also, note that I do not insist that the sensing task is individuated semantically. As far as I'm concerned at this point, the task can also be individuated non-semantically (e.g., as per Piccinini's proposal discussed in the next section). The issue here is whether we treat this *AND* explanation as a computational one even though there is more basic functional, tri-stable, structure.

One option is to treat the formal explanation as a computational explanation of the sensing task. This is the option that I favor. And I think that I'm in the same camp here as non-semanticists such as Egan and Piccinini, who treat such formal explanations as computational. But I guess that Coelho Mollo would reject this route. According to him, this *AND* explanation has nothing to do with computational individuation, which is more basic. Thus, unless he concedes that computational explanation is not related to computational individuation, Coelho Mollo would not treat the *AND* explanation as a computational one.

The other option is to deny that the *AND* explanation is a computational one. Computational explanations invoke the underlying functional profile and not the logical functions. Although certainly coherent, this view is unfaithful to computational explanations in the sciences. We take it that (computing) zero-crossing of second-derivative Laplacians explains the fact that the system performs edge-detection (Marr 1982), that (computing) integration explains the fact that the system produces signals of eye position (Robinson 1989), and that implementing *AND-XOR* (in Block's machine) explains the fact that the system performs addition (again, all these explanations are formal ones and have to be distinguished from explanations that refer to the specific contents of the states). These are all considered to be good computational explanations whose adequacy is independent of whether or not these systems have more basic functional profiles. Denying that these explanations are computational might leave us, at the end of the day, with a non-semantic notion of computation but one that is very limited in its scope.

Dewhurst has a somewhat different take on the issue of computational equivalence:

> Taken to its logical extreme, this argument might imply that no two systems are computationally equivalent. In practice, the physical structure of two computing mechanisms is always going to be distinct, and it is unclear whether we can draw any non-arbitrary boundary between the structures that are relevant or irrelevant to

> computational individuation. This is a serious issue, and at this point I am unsure how a proponent of the mechanistic account ought best to respond (2018: 110).

Dewhurst offers two lines of response. One is to bite the bullet at the risk of *reductio ad absurdum*. I will assume that biting the bullet is not a good option (why not adopt the semantic view instead?). A second line of response is to point out that this problem of equivalence confronts the mechanistic account in general and is not specific to computational mechanistic accounts. No two physical systems are equivalent in all respects. In practice, however, when we want to account for certain phenomenon, we appeal to experts who can identify the *relevant* properties of the system. They can tell which properties, e.g., temperature, are relevant for the individuation of these systems, and hence, for determining whether they are equivalent or not with respect to this (explanandum phenomenon). The same should be true for computing systems. It might be the case that no pair of sensors is physically equivalent. But we are still free to ask the experts which properties are relevant when accounting for these sensors. Confining ourselves to these relevant properties, we might find that our sensors are computationally equivalent, namely, that they have exactly the same relevant computational properties.

I agree that, when we explain a certain phenomenon, we take into account the properties that are (e.g., causally) relevant to the explanandum phenomenon. It is thus highly plausible that two physical systems that are not physically equivalent with respect to the totality of their physical properties are still equivalent with respect to some sub-class of their physical properties. A bar of metal and a bar of wood are different with respect to the totality of their physical properties as they are made of different materials. Still, the theorist can treat them as physically equivalent if they have the same temperature and their temperature is the factor relevant to the explanandum phenomenon. But how do these observations extend to the analogous computational case? What are the *relevant* computational properties that are analogous to the temperature of the bars? While our physicist can select from the reservoir of physical properties those that are relevant to the explanandum phenomenon, we need to know the analogous reservoir of computational properties.

Take two systems (e.g., sensors) with different basic computational structures. What would make them computationally equivalent? What is the reservoir of their potentially relevant computational properties? One option (advocated by Dewhurst and Coelho Mollo) is that the reservoir includes only one such property, namely, their basic (mechanistic/functional) structure. On these proposals, the two systems are not computationally equivalent *tout court*. If they don't share the basic structure, then they don't share computational properties at all. And if they don't share computational properties, they also don't share relevant computational properties. In other words, the appeal to relevance has no impact on computational equivalence. The two systems will be computationally distinct regardless of the explanandum phenomenon.

Another option is that the reservoir of computational properties includes more than one, and perhaps many, non-basic computational properties. This option seems to accord with the first (maximality) and last (entire span) basic-structure proposals. These proposals are consistent with the claim that the reservoir of the computational properties includes, in addition to the basic structure, also other non-maximal automata and logical functions. On this option, the two systems can be counted as computationally equivalent since they implement the same non-maximal automaton (or logical function). Yet this is precisely my claim! To recap, the objection was that computational individuation always takes into account a more basic structure. But the second option implies that computational individuation actually might not take into account the more basic structure. A computational individuation can take into account only one of the automata that the system implements, which is exactly my claim. We might yet disagree on whether the factors that affect the selection of the automaton are semantic or not (although I recall here that Dewhurst agrees that "the logical status of a gate is indeterminate prior to the attribution or identification of its representational content"). But this is a different issue; indeed, it is the subject of objection #2, to which we shall turn momentarily.

The upshot is that Dewhurst's appeal to relevance does not resolve the problem of computational equivalence. Either it has no effect on computational equivalence (as in Dewhurst's and Coelho Mollo's proposals) or it supports my claim that computational

individuation might take into account only one of the implemented automata – the one that is relevant to the explanandum phenomenon (as in the maximality and entire-span proposals).

5. Objection 2: Externalism without content

Gualtiero Piccinini (2008; 2015: 40-44) agrees that there is a further constraint that determines which automaton is the computational structure of a system in a certain context. He also agrees that this constraint takes into account features that are external to the computing systems. He denies, however, that these features must be semantic. In other words, Piccinini takes the master argument to establish externalism about computation, namely, the view that computational individuation essentially takes into account features that are external to the system (see also Horowitz 2007 and Shea 2013). But he refuses to accept the further step that these external features are semantic; he says they aren't: "Provided that the interaction between a mechanism and its context plays a role in individuating its functional (including computational) properties, a (non-semantic) functional individuation of computational states is sufficient to determine which task is being performed by a mechanism, and hence which computation is explanatory in a context" (2015: 43).

More specifically, Piccinini argues that the master argument rests on the premise that tasks (i.e., the explanandum input-output function) are individuated semantically. But, according to him, the semantic individuation of tasks (in these cases) does not entail the semantic individuation of computational states. Tasks are *also* individuated functionally (non-semantically), and it is this functional task description that is relevant for the computational individuation of the systems. Piccinini says that the proponents of the semantic account give no reason to prefer a semantic individuation of tasks to his wide functional individuation. In fact, he says, we can single out a functional individuation of a task that determines computational structure. Hence, we need not appeal to the semantic task for the individuation of computational structure.

As stated above, Piccinini concedes that we need to take into account the functional task in which computation is embedded, and that this task may depend on the interaction between

the computing mechanism and its context, e.g., on "which external events cause certain internal events" (2008: 220). Nonetheless, he argues that the environment need not be very wide. In artificial computing systems, the relevant functional properties might be input (e.g., keyboard) and output (e.g., display) devices. In the cognitive case, it might be sensory receptors and muscle fibers. This might be enough to determine whether a computation is being performed and, if so, which one. On this understanding, computational taxonomy selects the relevant automaton on the basis of the way the implementing mechanism interacts with events that are not part of the computing mechanism. These external features define the functional task, and it is the functional task and not the semantic one that is essential for computational individuation.

***Reply*:** Piccinini has a point, of course. The argument for premise 2 does not rule out a non-semantic individuation. The argument demonstrates how content can be used to decide which implemented automata are relevant to computational individuation.[24] But, if Piccinini shows that some functional "wider" factors (that define the functional task) suffice to define computational structure, then the argument for premise (2) fails. In this case, we can always individuate computational states without appealing to the content of the states. Thus, the way I see the dialectics of the debate is as follows: If the non-semanticist can present a credible non-semantic account that shows how to decide between the implemented automata, then the non-semantic view wins. As long as the non-semanticist does not come up with such an account, we have good reason to retain the semantic account.

Let us turn to the functional account proposed by Piccinini. The idea is to look at the interactions of the inputs and outputs with the surrounding external mechanism. Let us assume that our device projects to arm movement. Assume also that the behavior of our system, conjoined with the arm movement, is given by table 6. Outputting L or M produces no movement, whereas outputting H produces movement. Thus, taking into account the

[24] In the longer version of the argument, I contend that the obvious non-semantic alternatives fail in this task (Shagrir 2001).

interaction with movements gives us a reason to pick out the *AND* automaton, where one output is implemented in H and another output in L and M. This move certainly makes sense. The problem with this proposal, however, is that it removes some cases of indeterminacies but not others. In other words, while adding external constraints to the implementation relation removes some indeterminacies, others remain.

To see this, let us modify the example just a bit. Assume that the H outputs are now plugged to (physical) great-movement, the M outputs to (physical) medium-movement, and the L outputs to no-movement (Table 7). In this case, we can individuate movement (which is just '1') either to high-movement or to medium-movement-plus-high-movement. As table 7 shows, if we choose the first option, we end up with the *AND* automaton, and if we choose the second, we end up with the *OR* automaton. How would we now decide which functional kinds are relevant in singling out a computational structure? Appealing to movement is no longer helpful. We do not want to identify movement with specific physical, or even geometrical, properties, for reasons of multiple realization. In other contexts, we can have the same functional task even if greater movement is associated with different physical properties. And we cannot correlate the movement with the implemented automata or their outputs, as each implemented automaton is correlated with a different individuation of movement. Thus, in this case (and obviously in others as well), the appeal to external (non-semantic) short-arm factors does not help in choosing between the implemented *AND* and *OR* automata.[25]

One might suggest going even more external, to the outside environment. We could say, for example, that great-movement results in reaching apples, whereas medium-movement does not. How should we treat this proposal? Importantly, I am not saying that there cannot be a long-arm wide functional task – one that is extended all the way into the environment – that resolves all cases of indeterminacy. But before adopting this or any other account, we should be convinced that the account is credible. In other words, we should be sure that the account removes these indeterminacies. This is not a trivial task: The phenomenon of simultaneous

[25] One could suggest that the appropriate computational structure is neither *AND* nor *OR*, but the tri-stable structure. The point of the example, however, is that the proximal interactions do not remove all indeterminacies. To see this, one can expand the number of possible movements beyond three.

implementation is a special case of a broader phenomenon, namely, that we can group the same physical states into different formal types in different ways. The construction is just extended to include the wider functional facts. We did this exercise when extending the functional typing to external outputs within the embedding system (movements), and we could further extend it to functional typing that includes environmental factors. It might turn out, for example, that medium-movement results in reaching small apples. A functional account should show us how the wider context helps in avoiding indeterminacies without referring to the content of the system's states, namely, without assuming that the contents of the states are (e.g.,) apples and oranges (which will make it a semantic account). As far as I know, this has not been shown yet.

I have argued that the non-semanticists have not yet provided an account of computation that singles out the (correct) implemented automaton. The appeal to short-arm factors, as Piccinini proposes, removes some, but not all, indeterminacies. The appeal to long-arm, environmental, factors is a suggestive move, but more detail would be required to convince that it indeed removes other potential indeterminacies. In the current situation, I see no reason to abandon the semantic account, which provides a simple and elegant solution to the issue of indeterminacy, in favor of a non-semantic view.

I could stop with this conclusion, but I want to argue for more than that. I want to say that, at least in some cases, we would favor the semantic proposal over the functional proposal *even if the functional proposal could always single out one implemented automaton*. While the argument is brief and tentative, I believe that it nevertheless has some merit. So here goes: Piccinini presents a picture of semantic and functional tasks that compete over the impact on computational individuation, arguing that computational individuation acts in accordance with the functional task. How should we understand the relations between these semantic and functional tasks? One possibility is that the two tasks are co-extensive. Under this understanding, Piccinini's argument comes down to the claim that the non-semantic task *naturalizes* the semantic task. But I do not see how this claim challenges the semantic view. For one thing, I don't think that one can support the claim that content has been naturalized. Never

mind that no one has as yet naturalized mental and neural content.[26] But even if this task were to be accomplished, one has yet to show that the (derived) contents ascribed to my laptop can be naturalized as well. Second, and more importantly, the debate about the semantic view is not about naturalism. As noted in section 2, the semantic view is consistent with, but not committed to, the view that all computational contents can be naturalized. The debate between semantic and non-semantic views is on whether or not the identity conditions of computation must involve content. If the identity conditions always involve content – naturalized or not – then the semantic view wins; otherwise, it loses. Thus, if all that is said is that the semantic properties that essentially affect computational individuation can be further analyzed in non-semantic terms, then, as far as I can tell, the semantic view of computation has the upper hand.[27]

A second, more reasonable, possibility is that the two tasks are not co-extensive – that there is a discrepancy between the semantic and the functional tasks. Under this understanding, Piccinini's claim is that a computational taxonomy would side with the functional task. I would like to challenge this claim. Let us imagine a case of discrepancy (if there isn't one, then we are back to the first possibility). Take the tri-stable device that simultaneously implements *AND* and *OR*. Let us assume that outputting M-H encodes oranges and outputting L encodes apples. In this case, the semantic content (e.g., the orange-content and apple-content) implies that the relevant computational structure is the *OR*. As it also turns out, however, the functional task implies that the relevant computational structure is the *AND* (Table 8). The outputs of our detector project to certain motor devices, e.g., arm movements. Outputs of H produce movements, and outputs of L-M produce no movement (if one is so inclined, she can further assume that movement results in reaching the object and that no movement has no such result; she can also assume that our system has never produced M values so the conflict between the semantic and the functional tasks has never come to the surface. Our system

[26] Sprevak thus says that "many contemporary philosophers suspect that representation simply cannot be naturalized" (2013: 547).

[27] See also Dewhurst (2016), who notes that such a naturalistic move would be self-defeating to the non-semantic view, as it would remove "one of the primary motivations for giving a non-representational account of computation in the first place".

would still have produced M values had it encountered certain oranges, and this output would have resulted in no movement).

Which automaton counts for computational identity? I think that we would agree that if we wish to explain the semantic task, namely, how the system categorizes the stimuli to apples and oranges, we will take the *OR* and not the *AND*. The *OR* helps to explain how the system categorizes certain stimuli as apples and other stimuli as oranges. The *AND* is irrelevant to the explanation as its states are not matched with the vehicles of the apple-contents and orange-contents. Note that I do not deny that the *AND* structure might explain the wide non-semantic functional task. Nor do I deny that, in a different context with different contents, the computational structure might be the *AND* rather than the *OR*. I also agree that we can individuate the *OR* mechanism in non-semantic terms. The point of the latter example is just this: If the explanandum is the semantic task (and the contents are as described in the example), then the automaton that is relevant to the explanation is the *OR* and not the *AND*.

The non-semanticist thus faces a dilemma: On the one horn, she can admit that the *OR* explanation is a *computational* explanation of the semantic task. But in this case, it seems that the semantic task, and not the functional one, determines that the explanatorily relevant automaton is the *OR* and not the *AND*. Assuming that this determination also tells us something about computational individuation, we can conclude that content plays an essential role in computational individuation, namely, that it plays a role in determining that the computational structure of the system is OR.

On the other horn, the non-semanticist can deny that the *OR* explanation, though a formal explanation, is a *computational* one. She can say that computational explanations do not explain semantic tasks. Instead, they are (formal) explanations of non-semantic tasks. This means, more generally, that computational theories do not explain the fact that manufactured systems perform (e.g.) addition, namely, the task that is identified in terms of the numbers represented by digits. It also means that computational (formal) theories in cognitive neuroscience and in computer vision do not explain semantic tasks such as edge-detection, shape-from-shading, structure-from-motion and so on. This is a coherent view, but it comes

with a price: It is not consonant with the explanatory powers that scientists attribute to computational theories.

Piccinini seems to take the first horn. He agrees that the explanandum of computational explanations – at least in some cases – is a semantic task. He denies, however, that this premise leads to the semantic individuation of computational states. He argues that computational individuation is affected by the functional task. But my argument shows, to the contrary, that in cases of discrepancy a computational taxonomy would opt for the semantic individuative scheme.

The overall argument can be summarized as follows: If the semantic and functional tasks are co-extensive, then the functional task is just the semantic task, perhaps in a naturalized form. And if the semantic and functional tasks are not co-extensive (and assuming, with Piccinini, that computational explanations account for semantic tasks), then the computational individuation would be affected by the semantic task. One way or another, the semantic task affects, at least in some cases, computational individuation.

6. *Summary*

My aims were to explicate the semantic view of computation and to show that the master argument for it survives recent counter-arguments. The master argument relies on the phenomenon of simultaneous implementation and on the premise that content determines which automaton is relevant to computational individuation. One kind of counter-argument is that computational individuation is not forced to choose between the implemented automata. It always picks out a more (and the same) basic computational structure. My reply was that this move might undermine the notion of computational equivalence and thus make computational individuation explanatorily unhelpful for most purposes. Another kind of counter-argument is that computational individuation is forced to rely on extrinsic features but that those need not be semantic. My reply was that the semantic view better accounts for the individuation of these extrinsic features than the proposed non-semantic externalist views.

Acknowledgments

I am thankful to Arnon Levy, Gualtiero Piccinini, Nick Shea, Mark Sprevak and the members of the *Computability: Historical, Logical, and Philosophical Foundations* group at the Israeli Institute of Advances Studies (Jack Copeland, Eli Dresner, Nir Fresco, Carl Posy, Diane Proudfoot, Stewart Shapiro and Moshe Vardi) for stimulating discussion. An early version of the paper was presented at the *8th Quadrennial Fellows Conference* (Pittsburgh Center for Philosophy of Science) in Lund and at the *International Association for Computing and Philosophy* (IACAP 2016) in Ferrara. Thanks to the audiences for useful discussion. I am grateful to three reviewers of *Synthese* for their thorough reading and insightful comments. This research was supported by the Israel Science Foundation grant 830/18.

Oron Shagrir
Departments of Philosophy and Cognitive Science
The Hebrew University of Jerusalem
Mt. Scopus, Jerusalem 91905
Israel

References:

Block, N., 1987. Advertisement for a semantics for psychology. *Midwest studies in philosophy*, *10*: 615–678.

Block, N. 1995. The mind as the software of the brain. In D. Osherson, L. Gleitman, S. Kosslyn, E. Smith, E. and S. Sternberg (eds.), *An Invitation to Cognitive Science, Volume 3: Thinking* (2nd edition). MIT Press, pp. 377–425.

Bontly, T., 1998. Individualism and the nature of syntactic states. *The British journal for the philosophy of science*, *49:* 557–574.

Burge, T., 1986. Individualism and psychology. *The Philosophical Review*, *95*: 3–45.

Burge, T., 2010. *Origins of Objectivity*. Oxford University Press.

Chalmers, D. J., 1996. Does a rock implement every finite-state automaton? *Synthese*, *108*: 309–333.

Chalmers, D.J., 2011. A computational foundation for the study of cognition. *Journal of Cognitive Science*, *12*: 323–357.

Chrisley, R. L., 1994. Why everything doesn't realize every computation. *Minds and Machines*, *4*: 403–420.

Churchland, P.S. and Sejnowski, T.J., 1992. *The Computational Brain*. MIT Press.

Coelho Mollo, D., 2018. Functional individuation, mechanistic implementation: the proper way of seeing the mechanistic view of concrete computation, *Synthese*, *195*: 3477–3497.

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

Crane, T., 1990. The language of thought: No syntax without semantics. *Mind & Language*, *5*: 187–212.

Crane, T., 2015. *The Mechanical Mind: A Philosophical Introduction to Minds, Machines and Mental Representation* (3rd edition). Routledge.

Cummins, R.C., 1989. *Meaning and Mental Representation*. MIT Press.

Dewhurst, J., 2016. Review of *Physical Computation: A Mechanistic Account* by Gualtiero Piccinini. *Philosophical Psychology*, *29*: 795-797.

Dewhurst, J., 2018. Individuation without representation. *The British Journal for the Philosophy of Science*. *69*: 103-116

Dietrich, E., 1990. Computationalism. *Social Epistemology*, *4*: 135-154.

Dretske, F.I., 1988. *Explaining behavior: Reasons in a world of causes*. MIT press.

Egan, F., 1991. Must psychology be individualistic. *The Philosophical Review*, *100*: 179-203.

Egan, F., 1995. Computation and content. *The Philosophical Review*, *104*: 181-203.

Egan, F., 2010. Computational models: a modest role for content. *Studies in History and Philosophy of Science Part A*, *41*: 253-259.

Egan, F., 2012. Metaphysics and computational cognitive science: Let's not let the tail wag the dog. *Journal of Cognitive Science*, *13:* 39–49.

Fodor, J.A., 1975. *The Language of Thought*. Harvard University Press.

Fodor, J.A., 1980. Methodological solipsism considered as a research strategy in cognitive psychology. *Behavioral and Brain Sciences*, *3*: 63-73.

Fodor, J.A., 1994. *The elm and the expert: Mentalese and its semantics*. MIT press.

Fresco, N., 2014. *Physical Computation and Cognitive Science*. Springer.

Fresco, N., Copeland, B.J. and Wolf, M., forthcoming. The indeterminacy of computation.

Godfrey-Smith, P., 2009. Triviality arguments against functionalism. *Philosophical Studies*, *145:* 273-295.

Horowitz, A., 2007. Computation, external factors, and cognitive explanations. *Philosophical Psychology*, *20*: 65-80.

Klein, C., 2008. Dispositional implementation solves the superfluous structure problem. *Synthese*, *165*: 141–153.

Ladyman, J., 2009. What does it mean to say that a physical system implements a computation? *Theoretical Computer Science*, *410*: 376-383.

Lee, J. forthcoming. Mechanisms, wide functions and content: Towards a computational pluralism. *The British Journal for the Philosophy of Science*.

Marr, D., 1982. *Vision: A Computational Investigation into the Human Representation and Processing of Visual Information*. Freeman and Company.

Matthews, R. J. and Dresner, E., 2016. Measurement and computational skepticism. *Noûs*, *51*: 832–854.

Melnyk, A., 1996. Searle's abstract argument against strong AI. *Synthese*, *108*: 391–419.

Miłkowski, M., 2013. *Explaining the Computational Mind*. MIT Press.

Millikan, R.G., 1984. *Language, thought, and other biological categories: New foundations for realism*. MIT press.

Minsky, M.L., 1967. *Computation: Finite and Infinite Machines*. Prentice-Hall.

Piccinini, G., 2008. Computation without representation. *Philosophical Studies*, *137*: 205–241.

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

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

Putnam, H., 1988. *Representation and Reality*. MIT Press.

Pylyshyn, Z.W., 1984. *Computation and Cognition: Toward a Foundation for Cognitive Science*. MIT Press.

Ramsey, W.M., 2007. *Representation Reconsidered*. Cambridge University Press.

Rescorla, M., 2012. Are computational transitions sensitive to semantics? *Australasian Journal of Philosophy*, *90*: 703-721.

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

Rescorla, M., 2014. A theory of computational implementation. *Synthese*, *191*: 1277-1307.

Rescorla, M. 2015. The computational theory of mind, *The Stanford Encyclopedia of Philosophy*, Edward N. Zalta (ed.), URL = <https://plato.stanford.edu/archives/spr2017/entries/computational-mind/>.

Robinson, D.A., 1989. Integrating with neurons. *Annual review of neuroscience*, *12:* 33-45.

Scheutz, M., 1999. When physical systems realize functions. *Minds and Machines*, *9*: 161–196.

Scheutz, M., 2001. Computational versus causal complexity. *Minds and Machines*, *11*: 543–566.

Scheutz, M., 2012. What it is not to implement a computation: a critical analysis of Chalmers' notion of implementation. *Journal of Cognitive Science*, *13*: 75–106.

Searle, J. R., 1992. *The Rediscovery of the Mind*. MIT Press.

Shagrir, O., 2001. Content, Computation and Externalism. *Mind*, *110*: 369-400.

Shagrir, O., 2006. Why We View the Brain as a Computer. *Synthese, 153*: 393–416.

Shea, N., 2013. Naturalising representational content. *Philosophy Compass*, *8*: 496-509.

Shea, N. *Representation in the Brain*. Forthcoming

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

Sprevak, M., 2013. Fictionalism about neural representations. *The Monist*, *96*: 539-560.

Stich, S.P., 1983. *From folk psychology to cognitive science: The case against belief*. MIT press.

Wilson, R.A., 1994. Wide computationalism. *Mind*, *103*: 351-372.

Input 1	Input 2	Output
5-10 V (H)	5-10 V (H)	5-10 V (H)
5-10 V (H)	2.5-5 V (M)	2.5-5 V (M)
5- 10 V (H)	0-2.5 V (L)	2.5-5 V (M)
2.5-5 V (M)	5-10 V (H)	2.5-5 V (M)
2.5-5 V (M)	2.5-5 V (M)	2.5-5 V (M)
2.5-5 V (M)	0-2.5 V (L)	2.5-5 V (M)
0-2.5 V (L)	5-10 V (H)	2.5-5 V (M)
0-2.5 V (L)	2.5-5 V (M)	2.5-5 V (M)
0-2.5 V (L)	0 – 2.5 V (L)	0-2.5 V (L)

Table 1: Physical gate **P**: The gate maps voltages from two input channels, Input 1 and Input 2, into one Output channel.

Input 1	Input 2	Output
1	1	1
1	0	0
0	1	0
0	0	0

Table 2: **P** implements *AND*: Under the assignment of '1' to H (5-10V) and '0' to L-M (0-5V), **P** implements *AND*.

Input 1	Input 2	Output
1	1	1
1	0	1
0	1	1
0	0	0

Table 3: **P** implements *OR*: Under the assignment of '1' to M-H (2.5-10V) and '0' to L (0-2.5V), **P** implements *OR*.

Input 1	Input 2	Output
H	H	H
H	M	M
H	L	L
M	H	M
M	M	M
M	L	L
L	H	L
L	M	L
L	L	L

Table 4: Physical gate **Q.** Under the assignment of '1' to H and '0' to L-M, **Q** implements *AND*. Under the assignment of '1' to M-H and '0' to L, **Q** implements *AND* (again).

Input 1	Input 2	Output
A	A	A
A	B	B
A	C	B
B	A	B
B	B	B
B	C	B
C	A	B
C	B	B
C	C	C

Table 5: The "maximal" automaton implemented by physical gate **P**. The labels A, B and C stand for different equivalence classes of inputs and outputs. The maximal automaton implies *AND* under the assignments of 1s to A and 0s to B and C. It implies *OR* under the assignments of 1s to A and B and 0s to C.

Input 1	Input 2	Output	Motor output
H	H	H	Movement
H	M	M	No movement
H	L	M	No movement
M	H	M	No movement
M	M	M	No movement
M	L	M	No movement
L	H	M	No movement
L	M	M	No movement
L	L	L	No movement

Table 6: Our toy-example automaton interacts with arm movement. Outputting H results in movement; outputting below L-M results in no movement.

Input 1	Input 2	Output	Motor output
H	H	H	High movement
H	M	M	Medium movement
H	L	M	Medium movement
M	H	M	Medium movement
M	M	M	Medium movement
M	L	M	Medium movement
L	H	M	Medium movement
L	M	M	Medium movement
L	L	L	No movement

Table 7: Our toy-example automaton interacts with arm movement. In this scenario, outputting H results in high movement, outputting M results in medium movement and outputting L results in no movement.

Input 1	**Input 2**	**Output**	Semantic output	Motor output
H	H	H	Orange content	Movement
H	M	M	Orange content	No movement
H	L	M	Orange content	No movement
M	H	M	Orange content	No movement
M	M	M	Orange content	No movement
M	L	M	Orange content	No movement
L	H	M	Orange content	No movement
L	M	M	Orange content	No movement
L	L	L	Apple content	No movement

Table 8: A discrepancy between semantic and functional tasks. The semantic task is correlated with groupings of L and M-H. The functional task is correlated with groupings of L-M and H.