**Mechanistic Computational Individuation without Biting the Bullet**

**Nir Fresco & Marcin Miłkowski**

**Abstract**

Is the mathematical function being computed by a given physical system determined by the system’s dynamics? This question is at the heart of the indeterminacy of computation phenomenon (Fresco *et al.* [unpublished]). A paradigmatic example is a conventional electrical AND-gate that is often said to compute conjunction, but it can just as well be used to compute disjunction. Despite the pervasiveness of this phenomenon in physical computational systems, it has been discussed in the philosophical literature only indirectly, mostly with reference to the debate over realism about physical computation and computationalism. A welcome exception is Dewhurst’s ([2018]) recent analysis of computational individuation under the mechanistic framework. He rejects the idea of appealing to semantic properties for determining the computational identity of a physical system. But Dewhurst seems to be too quick to pay the price of giving up the notion of *computational equivalence*. We aim to show that the mechanist need not pay this price. The mechanistic framework can, in principle, preserve the idea of computational equivalence even between two different enough kinds of physical systems, say, electrical and hydraulic ones.

1. *Discussion*
2. *Conclusion*
3. **Discussion**

Although the topic of computation—as it pertains to computational explanation and the mechanistic approach to cognitive theorising—has been the centre of much philosophical discussion, the phenomenon of indeterminacy of computation has been given less attention (Dennett [1978]; Block [1990]; Putnam [1991]; Searle [1992], p. 199; Shagrir [2001], [2012]; Buechner [2008]; Bishop [2009]; Sprevak [2010]; Fresco [2015]; Piccinini [2015]; Coelho Mollo [2018]; Miłkowski [2017]). It has been mostly considered a side issue in the debate over whether there are any facts of the matter about what a given physical system computes and its implications for computationalism. The phenomenon itself has been often considered by many as problematic, and even detrimental to computational analyses of cognitive systems (see, for example, Bishop [2009]; Sprevak [2010]; Shagrir [2012]). In the debate on the metaphysical nature of computation, whether or not a given account of computation is able to “resolve” the indeterminacy of computation problem has been considered a litmus test for its adequacy. The notorious case study from Boolean engineering for analysing this phenomenon has been the duality of two simple Boolean gates: a two-input, single-output AND-gate and a two-input, single-output OR-gate. There is nothing special about these two gates: the phenomenon occurs in twelve out of sixteen two-input, single-output Boolean gates (XOR and XNOR, NAND and NOR, etc.).

Two noteworthy, recent discussions of this phenomenon are due to Dewhurst ([2018]) and Lee ([2018]), both of which centre on the question of computational individuation. They also share a similar strategy for dealing with this question. Namely, they attempt to show how the mechanistic account of computation can face the challenge mounted by the computational semanticist (see, for example, Bishop [2009]; Sprevak [2010]; Shagrir [2012]). The semanticist argues that we must appeal to semantic (or representational) properties in order to individuate computations when indeterminacy “threatens” (as in the case of a physical device that gives rise either to conjunction or disjunction). We will focus here on Dewhurst’s analysis, and more specifically on the concession he is willing to make, on behalf of the mechanist, though it is not clear that such concession is needed.

Dewhurst builds on Piccinini’s mechanistic account of computation to provide a non-semantic individuation of digits and processing units—two key ingredients in the decomposition of physical computational systems (Dewhurst [2016]). Dewhurst claims that prior to assigning semantic content to the digits processed by a physical device X (see Tab. 1), one cannot tell whether this device is actually an AND-gate or an OR-gate. However, he does add that this device can be distinguished from another device Y (see Tab. 2). This distinction can be accomplished, so he claims, by simply describing the physical, systematic behaviour of the device in terms of its various components (that is, digits and processing units), and how they function *within* an *encompassing* system.

| **Input-1 (Voltage)** | **Input-2 (Voltage)** | **Output (Voltage)** |
| --- | --- | --- |
| 0.1-0.5 | 0.1-0.5 | 0.1-0.5 |
| 0.1-0.5 | 0.6-1.0 | 0.1-0.5 |
| 0.6-1.0 | 0.1-0.5 | 0.1-0.5 |
| 0.6-1.0 | 0.6-1.0 | 0.6-1.0 |

**Table 1**. A description of a physical device, X, in terms of its possible inputs and corresponding outputs in terms of voltage levels. The device operates on two voltage ranges, low: 0.1v-0.5v, and high: 0.6v-1.0v.

| **Input-1 (Voltage)** | **Input-2 (Voltage)** | **Output (Voltage)** |
| --- | --- | --- |
| 0.1-0.5 | 0.1-0.5 | 0.1-0.5 |
| 0.1-0.5 | 0.6-1.0 | 0.6-1.0 |
| 0.6-1.0 | 0.1-0.5 | 0.6-1.0 |
| 0.6-1.0 | 0.6-1.0 | 0.6-1.0 |

**Table 2**. A description of a physical device, Y, in terms of its possible inputs and corresponding outputs in terms of voltage levels. Notice the different input/output relations compared with device X in the two middle rows.

Dewhurst even agrees that, when considered in isolation, either of these two devices could be used by an encompassing system to perform *both* logical operations. He refers to such operations as “algorithmic”. A device typically used in computer engineering for computing conjunction (when the low voltage range is interpreted as false, and the high voltage range as true) can in another architectural setup (where the low voltage range is interpreted as true, and the high voltage range as false) be used for computing disjunction. Note that the device itself does not change, only the setup of the encompassing system, which uses it in a specific way, does. So far so good. But the next proposal leads Dewhurst to a concession he need not make; at least not according to the mechanistic framework that is adopted.

Dewhurst proposes that placing devices X and Y in the same encompassing system is *sufficient* for discriminating between them. In line with the mechanistic framework, the physical mechanisms concerned suffice for the computational individuation of devices X and Y: *pace* the semanticist, no assignment of semantic properties is required for this task. The mechanist can specify the computational identity of a given system in virtue of (1) the number of unique digits processed, (2) the number of processing units, and (3) the input-output relations in which the digits partake in the encompassing system (to be sure, the relevant input-output relations are those between the part-system concerned and the encompassing system).

Let us now consider in more detail how devices X and Y can be individuated mechanistically. Both process two distinct digits: one digit is the low voltage range, the other is the high voltage range. We do not need to assign any content to these digits, but just note that they are distinct. Since the two devices process the same number of digits, they cannot be simply computationally individuated on that ground. They also have the same number of processing units, namely one. Thus, this fact of the matter, too, is insufficient for the computational individuation of devices X and Y. The third fact of the matter concerning these devices is their input-output relations exploited by the encompassing system. Here the different input-output relations *can be* used for determining the distinct computational identity of these devices *within the encompassing system*. On the other hand, if we were to computationally individuate two systems that *differ* in the number of processing units or digits (as in Shagrir’s tri-stable vs. bi-stable computational system ([2012])), that fact of matter would suffice for distinguishing between them computationally—even when the two systems performed the same *mathematical function*.

Dewhurst proposes to keep the notions of *algorithmic equivalence* and *computational equivalence* distinct. Two algorithmically equivalent systems (say, AND-gates) can nevertheless be computationally distinct (say, if one processed three digits and the other only two). This is an important feature of computational systems, so Dewhurst tells us: two computationally *distinct* systems may be used to compute the *same* mathematical function. But then he adds surprisingly that ‘[t]aken to its logical extreme this […] might imply that no two systems are computationally equivalent. In practice the physical structure of two computing mechanisms is always going to be distinct’ (Dewhurst [2018], p. 110). The implications of these two claims are rather problematic, but what is not clear to us is why Dewhurst should be willing to make this concession.

True: any two physical devices will inevitably be physically different, no matter how minute the differences. Consider again device X, which is described in Tab. 1, and suppose further that the operational voltage ranges of its counterpart, device X’, are rather (0.11v—0.5v) and (0.61v—1.0v). Dewhurst seems to suggest that although both devices X and X’ may give rise—at the “algorithmic level”—to conjunction, they are computationally distinct. For the low range of X’ differs from that of X by 0.01v and similarly for the high range of X’ compared with that of X. Of course, the differences in question may be arbitrarily small or large. And if that were the case, then electrical AND-gates and hydraulic AND-gates, for example, would be likewise computationally distinct. This strikes us as at least counterintuitive, if not outright false to claim that *any* two AND-gates are not computationally equivalent. If that were indeed an implication of the mechanistic view, its opponents could rightly point this out to argue against that view.

However, the mechanist need not bite this bullet. By Dewhurst’s lights, device X operates on two digits and so does device X’. Why should the digits processed by X and those processed by X’ be distinct, according to the mechanistic account of computation? Compare that with device Y, which is described in Tab. 2. Device X operates on two digits and so does device Y. All that matters is that devices X and Y process two unique digits each. But—as Dewhurst tells us—devices X and Y are computationally distinct, because the input-output relations in which the digits partake in device X differ from those in device Y relative to the encompassing system. Returning to devices X and X’, each device processes two digits, and the input-output relations in which the digits partake are equivalent. The number of processing units in devices X and X’ is also the same: precisely one. Therefore, the mechanist may safely conclude that devices X and X’ are computationally identical.

Do these individuation criteria introduce some ‘minimal semantic content’ due to the appeal to the way digits are treated by the encompassing system? And if so, is that problematic? The mechanist maintains that computation can, but *need not*, be semantic. Our proposal is compatible, on the one hand, with some form of *denotational* (or *procedural*) semantics. For example, the primitives of a computer language can be interpreted as actions on the, somewhat abstracted, internal state of the computing system (White [2011], p. 194). This idea resembles Piccinini’s notion of ‘internal semantics’ according to which contents are assigned to the computing system’s internal components and activities ([2015], p. 46). On the other hand, our proposal is compatible with consumer-based teleosemantics. Whether it is device X or device Y that is taken as an AND-gate (or, equivalently, an OR-gate) depends, at least, in part on how the consuming system uses it. Similarly, whether devices X and X’ are computationally equivalent depends, at least in part, on whether the encompassing system is sensitive to the differences between the corresponding low and high voltage ranges. Whichever semantics is implicated here[[1]](#footnote-1) it is internalised and innocuous enough. Importantly, this kind of semantics does not solve, for example, the Symbol Grounding Problem (Harnad [1990]). It does not specify what it takes for a computational vehicle to stand for anything outside the immediate realm of the computing system.

Our proposal about computational equivalence is not restricted to physical systems of the same kind, for example, only *electrical* Boolean gates and circuits. Rather, we maintain that electrical AND-gates and hydraulic AND-gates, for example, can be likewise considered computationally equivalent. A hydraulic device (a) in which two digits are processed, (b) consists of a single processing unit, and (c) whose relevant input-output relations in which the digits partake correspond to conjunction is computationally equivalent to device X.[[2]](#footnote-2) Importantly, one should distinguish computational equivalence from what Wimsatt ([2002]) calls ‘functional isomorphism under substitution’. He rightly stresses that a significant effort is required to preserve compatibility between computational systems. Two computers may run exactly the same kind of software without the intersubstitutability of their electrical components. For example, IBM 709 and IBM 7090 shared exactly the same circuit logic diagrams, but one was made of tubes and the other of transistors.

Therefore, the computational equivalence of two different computing systems (such as computers or Boolean gates) violates Schiller’s ([2018]) strong version of the ‘swapping constraint’. Roughly, according to this constraint, two physical components (‘states’ in his proposal) are computationally equivalent when swapping them in an encompassing system does not compromise the system’s computational integrity. The strong version of Schiller’s constraint also states that such a “harmless” swap would occur *without* any further transduction changes.[[3]](#footnote-3) However, realistically one cannot simply swap, say, a transistor in an (transistor-based) IBM 7090 with a tube and expect the computer to work as before (that is, its computational integrity remaining intact). Nevertheless, the addition of suitable transduction of signals into the same type of physical vehicle allows even more varied kinds of computing systems to exchange the relevant information, which may be transmitted by different physical media (Paul [2006]). In a nutshell, a suitable transduction of signals into a single type of physical medium between two computing systems A and B may lead to A and B remaining computationally equivalent. Thus, an electrical AND-gate *can be* deemed computationally equivalent to a hydraulic one. Our proposal is consistent with the *weak* version of Schiller’s constraint, which allows for adjustments in the transducer layer of a system and yet is rejected by Schiller.

1. **Conclusion**

If our analysis is right, then the mechanistic account remains a plausible alternative to semantic views of computation. The latter rely on the indeterminacy phenomenon to argue against mechanistic views of computation (for example, Shagrir [2018]). However, *contra* Dewhurst and Shagrir, the mechanistic view need not undermine the notion of computational equivalence. We have argued, on mechanistic grounds, that two physical systems are computationally equivalent as long as they process the same number of digits, are based on the same number of processing units, and preserve the input-output relations between the digits. Our proposal is not limited to within-system computational equivalence, but it rather extends to different enough kinds of physical systems, so long as they satisfy specific conditions. Nevertheless, Dewhurst’s algorithmic individuation may still be indeterminate (as in the case of devices X and X’ described above): two devices may be computationally equivalent, yet each of them may be used to compute two (or more) functions (such as, conjunction or disjunction).

**Acknowledgments**

The authors thank Marty J. Wolf for comments on an earlier draft of this manuscript. This research was kindly supported by the Polish Academy of Science (funding for the study visit of Nir Fresco in Warsaw in 2018) as well as by National Science Centre, Poland (grant no. 2014/14/E/HS1/00803, PI: Marcin Miłkowski) and Israel Science Foundation (grant no. 830/18, PI: Oron Shagrir).

*Nir Fresco*

*ORCID: 0000-0002-7182-9727*

*The Department of Brain & Cognitive Sciences*

*The Department of Philosophy*

*Ben-Gurion University of the Negev*

*Beer Sheva, Israel*

[*nfresco@bgu.ac.il*](mailto:nfresco@bgu.ac.il)

*Marcin Miłkowski*

*ORCID: 0000-0001-7646-5742*

*Section of Logic & Cognitive Science*

*Institute of Philosophy and Sociology*

*Polish Academy of Sciences*

*Warsaw, Poland*

[*Marcin.Milkowski@ifispan.waw.pl*](mailto:Marcin.Milkowski@ifispan.waw.pl)

**References**

Bishop, J. M. [2009]: ‘A Cognitive Computation Fallacy? Cognition, Computations and Panpsychism’, *Cognitive Computation*, **1**, pp. 221–33.

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, pp. 137–70.

Buechner, J. [2008]: *Godel, Putnam, and Functionalism: A New Reading of Representation and Reality*, Cambridge: The MIT Press.

Coelho Mollo, D. [2018]: ‘Functional Individuation, Mechanistic Implementation: The Proper Way of Seeing the Mechanistic View of Concrete Computation’, *Synthese*, **195**, pp. 3477–97.

Dennett, D. C. [1978]: *Brainstorms: Philosophical Essays on Mind and Psychology*, Cambridge: The MIT Press.

Dewhurst, J. [2018]: ‘Individuation without Representation’, *The British Journal for the Philosophy of Science*, **69**, pp. 103–16.

Fresco, N. [2015]: ‘Objective Computation Versus Subjective Computation’, *Erkenntnis*, **80**, pp. 1031–53.

Fresco, N., Wolf, M. J., & Copeland, B. J. (unpublished). ‘The Indeterminacy of Computation’.

Harnad, S. [1990]: ‘The Symbol Grounding Problem.’ *Physica D* **42**, pp. 335–46.

Lee, J. [2018]: ‘Mechanisms, Wide Functions, and Content: Towards a Computational Pluralism’, *The British Journal for the Philosophy of Science*. DOI: 10.1093/bjps/axy061.

Miłkowski, M. [2017]: ‘The False Dichotomy Between Causal Realization and Semantic Computation’, *Hybris*, **38**, pp. 1-21.

Paul, C. [2006]: ‘Morphological Computation. A Basis for the Analysis of Morphology and Control Requirements’, *Robotics and Autonomous Systems*, **54**, pp. 619–30.

Piccinini, G. [2015]: ‘Physical Computation: A Mechanistic Account’, New York, NY: Oxford University Press.

Putnam, H. [1991]: *Representation and Reality*, Cambridge: The MIT Press.

Schiller, H. I. [2018]: ‘The Swapping Constraint’. *Minds and Machines*, ***28***, 605–622.

Searle, J. R. [1992]: *The Rediscovery of the Mind*, Cambridge: The MIT Press.

Shagrir, O. [2001]: ‘Content, Computation and Externalism’, *Mind*, **110**, pp. 369–400.

——— [2012]: ‘Computation, Implementation, Cognition’, *Minds and Machines*, **22**, pp. 137–48.

——— [2018]: ‘In Defense of the Semantic View of Computation’, *Synthese*, DOI: 10.1007/s11229-018-01921-z.

Sprevak, M. [2010]: ‘Computation, Individuation, and the Received View on Representation’, *Studies in History and Philosophy of Science Part A*, **41**, pp. 260–70.

Wimsatt, W. C. [2002]: ‘Functional Organization, Analogy, and Inference’, in A. Ariew, R. Cummins and M. Perlman (*eds*), *Functions: New Essays in the Philosophy of Psychology and Biology*, Oxford: Oxford University Press, pp. 173–221.

1. Space considerations prevent us from discussing this important issue in greater detail. [↑](#footnote-ref-1)
2. Coelho Mollo ([2018], sec. 7) claims that a hydraulic device would be computationally equivalent to an electrical AND-gate, if the former is both sensitive to and responds in the same (uniform) way to the same number of what he calls ‘equivalence classes’ of physical states. We lack the space to discuss his suggestion here. [↑](#footnote-ref-2)
3. Transduction changes may be required to accommodate physical changes in the swapped components, such as sensitivity to different voltage ranges. [↑](#footnote-ref-3)