Rejecting the Received View: Representation, Computation, and Observer-Relativity

Joe Dewhurst[1]

Abstract. I defend Piccinini's mechanistic account of computation against three related criticisms adapted from Sprevak's critique of non-representational computation. I then argue that this defence highlights a major problem with what Sprevak calls the received view; namely, that representation introduces observer-relativity into our account of computation. I conclude that if we want to retain an objective account of computation, we should reject the received view.

1 INTRODUCTION

Sprevak [1] outlines what he calls "the received view" of computation, which is apparently held by a majority of both philosophical and scientific experts on computation. As he puts it, this is the view that "a necessary condition on any physical process counting as a computation is that it possesses representational content" [1, p. 260]. He clarifies that what he has in mind is a very minimal form of representation, involving no more than "a basic notion of aboutness or reference [that] should link an entity and a content" [1, p. 261]. Specifically, he wants to avoid any mention of the kind of intentional content that might "plausibly require the involvement of cognitive agents" [1, p. 261]. Furthermore, it is in terms of *individuation* rather than *causal dynamics* that representation should be understood as necessary for computation [1, pp. 261-2]. This is meant to avoid a line of criticism regarding causal irrelevance that I will not pursue here.

Thus the "battleground", as Sprevak puts it, is whether or not representation is necessary for computational individuation. There have been several recent attempts at giving alternative, non-representational, accounts of computation. I will focus on Piccinini's account, as I feel that it helps elucidate some of the main problems with the received view. Specifically, it points towards the introduction of observer-relativity as a result of the received view's representation requirement.

In brief, the mechanistic account claims that computation consists solely in the manipulation of strings of digits by a series of processors, and that both digits and processors can be given a non-representational specification [2, pp. 508-12]. Digits are individuated functionally, such that every digit of the same type is manipulated in the same way by any given processor-type. Similarly, processors are individuated in terms of their interaction with digits. What makes these digits and processors non-representational is that they function purely by virtue of their physical interactions within the mechanism, and require no further level of analysis. Whilst they can be given a representational interpretation, this is not intrinsic to either their mechanistic role or their individuation (see [3]).

In addition to the processor there might also be an input device (to turn external stimuli into digits), a memory device (to store strings of digits whilst they are not in use), and an output device (to turn digits into a suitable output) [2, p. 514]. None of these components are strictly necessary for computation, but it would be unusual to find a computing mechanism that did not include them.

In defending the received view, Sprevak gives three related arguments against non-representational computation. His arguments are primarily aimed at Egan's mathematical theory of computation, but he indicates that similar criticisms might be made of the mechanistic account [1, p. 260]. I will reconstruct each argument so as to apply directly to the mechanistic account, and attempt to respond to them by clarifying certain key aspects of the account.

2 PARADIGMATIC CASES

Sprevak's initial move is to draw attention to the role that representational content appears to play in paradigmatic cases of computation. He cites two cases: "Turing's mindless clerk" who performs a mapping from ink-mark representations to other ink-mark representations; and the electrical signals within an electronic computer, which he claims represent both 0s and 1s as well as whatever textual or pictorial information the computer is processing [1, p. 267]. In both cases he notes that there is an ambiguity between the representation of abstract numbers and the representation of meaningful content. I will assume that either outcome would be sufficient to refute the claim that mechanistic computation is non-representational.

Sprevak admits that the apparently representational nature of these paradigmatic cases of computation is "far from conclusive", but he takes it to be at least indicative of the role that representational content plays in computation [1, p. 268]. He argues that we should "keep the paradigmatic cases in mind" when it comes to judging whether or not a given system is computational [1, p. 268]. I agree that this is a viable tactic, and we should note that it is one of Piccinini's criteria for assessing accounts of computation [2, p. 502]. However it is only one criterion out of six, and as such it should not be considered the final arbiter of what constitutes computation. The mechanistic account is intended to give independent grounds for assessing computation, and crucially it aims to remain neutral on questions of representational content. To presume that representational content is essential to computation because of these paradigmatic cases would beg the question against the mechanistic account.

It might turn out that all cases of physical computation do involve representational content, but if the mechanistic

[1] Dept. of Philosophy, Univ. of Edinburgh, EH3 9AD, UK. Email: joseph.e.dewhurst@gmail.com

account is correct this would not be a feature of their being computational systems, but rather a feature of whatever theory of representational content we think is true. The question at hand is not whether computational systems ever possess representational content, but rather whether representational content is a necessary requirement for something to be considered a computational system. The two cases that Sprevak mentions indicate that the former might be true, but say nothing conclusive about the latter.

Both cases also rest on the supposedly representational nature of two further features of computation: input/output mapping and the individuation of digits. Sprevak presents arguments for each, which I will consider below. If these features turn out not to be representational, then the argument from paradigmatic cases will be undermined, or at least weakened. Without further support, our intuitions alone cannot conclusively prove that computation is representational.

3 INPUT/OUPUT MAPPING

Sprevak's second argument provides an expanded analysis of the input and output components in computing mechanisms. This argument comes in two parts, which I will present and respond to in turn.

As indicated above, paradigmatic cases of computation are often described as transforming representational inputs into representational outputs. It is Sprevak's contention that this description, and the associated representational content of computational states, is essential in order to arbitrate the status of physically divergent computational systems.

In such cases, where two systems are physically very different, it is a common strategy to say that they are computing the same function if their inputs and outputs are representationally equivalent. We cannot simply appeal to the physical equivalence of the inputs and outputs, for this will differ along with the systems themselves. For example, two systems might perform the same calculation, but one could take electrical signals as inputs and outputs, and the other could take marbles [1, p. 268]. Sprevak's claim is that the reason we can identify them as performing the same computation despite the difference in input and output is that both the marbles and the electrical signals have the same representational content. Therefore, physical computation must be understood as inherently representational in order to explain this kind of input/output equivalence.

Under the mechanistic account, computational systems are differentiated in terms of the functional structure of their components. This means that two physically divergent systems can be said to perform the same computation if and only if they possess a relevantly similar structure. Of primary importance is the organisation of the components that serve as digits and processors. The input and output components are only of secondary importance, insofar as they produce or receive strings of digits. Hence whether a system's input comes in the form of electrical signals or marbles is irrelevant to the computation that it performs, which will be characterised solely in terms of the processing of strings of digits.

As a result of this, the mechanistic account cannot accommodate input/output equivalence in the sense that Sprevak describes it. What makes input and output components equivalent is their causal relationship to the internal structure of the computing mechanism, rather than their relationship with the external world. Regardless of whether or not we map the same content to electrical signals and marbles, the mechanisms that they interact with will only be computationally equivalent if the structure of their digits and processors are equivalent. This means that, perhaps contrary to our intuitions, two systems that possess representationally equivalent input/output mappings might nonetheless be computationally distinct.

Sprevak's second illustration makes the point clear. Consider two computational systems, identical aside from their inputs and outputs. One processes chess moves, and the other makes stock market predictions. Their inputs and outputs appear to differ, but there is a sense in which we want to say that they are the same. As Sprevak puts it,

> [w]e seem inclined to say that, in a sense, the two processes compute different functions, yet in another sense they are I/O equivalent. [1, p. 268]

"Appeal to representational content", he continues, "can accommodate both judgements" [1, p. 268]. They differ insofar as the content of their inputs and outputs differ, but they are equivalent because we can "*easily interpret* the two processes so that they compute the same function" [1, p. 268, emphasis in original]. Without the appeal to representational content, we would apparently be unable to reconcile these two intuitions.

The mechanistic account is only able to accommodate the intuition that they are computing the same function. The chess-playing computer and the stock-market-predicting computer will be equivalent if and only if their internal structures are equivalent. Insofar as they are processing digits in the same way, the two systems are performing the same computations. The implausibility of this scenario enhances our intuition that the computations they are performing must be different, but the nature of their inputs and outputs is irrelevant to their computational equivalence. What use we put them to, and what content they come to possess as a result of this, is strictly irrelevant to their status as computing mechanisms.

Quite aside from being a weakness, this failure to accommodate both intuitions indicates the capacity for the mechanistic account to provide determinate answers in cases such as this. Our intuitions may differ, but we should not always trust intuitions, and the mechanistic account enables us to appeal to an objective standard in determining the equivalence (or not) of computational systems.

My response to this argument is reflected in Piccinini's original formulation of the computing mechanism, which consists only of a string of digits and a processor [2, pp. 508-12]. He notes that in practice computing mechanisms will typically have additional components, including input and output devices, but these are not essential to the basic functional description [2, p. 514]. We can individuate computational states without making reference to inputs and outputs, and are therefore able to distinguish between equivalent and non-equivalent computational systems without being misled by our intuitions about representational content.

It should be reiterated here that the claim is not that the representational content of inputs and outputs is completely irrelevant, but only that it is irrelevant insofar as computation itself is concerned. When it comes to attributing representational

content to computational states, we will inevitably look to the content of the inputs and outputs to guide us. This is why we differentiate between the chess-playing and stock-market-predicting computers, and why in representational terms they are non-equivalent. The point is that representational non-equivalence should not be equated with computational non-equivalence – the two systems are computationally equivalent whilst possessing distinct representational content. Hence we come to have divergent intuitions about whether or not they are equivalent, because one intuition is based on our understanding of computation, and the other on our understanding of representational content. In demonstrating this, and clarifying what it means for two systems to be computationally equivalent, the mechanistic account of computation has done us a valuable service. Equating input/output equivalence with computational equivalence might have historically been a useful heuristic, but it distracts us from what is genuinely important about computational systems: the internal processing of digits, rather than the external relationship between inputs and outputs.

4 INDIVIDUATING DIGITS

Sprevak's third argument is that in order to distinguish digits we have to recognise that they possess at least a form of minimal content. He presents the pair of tables that specify an AND gate and an OR gate:

> The output of an AND gate is *1* just in case both inputs are *1* otherwise it is *0*. The output of an OR gate is *0* just in case both inputs are *0* otherwise it is *1*. [1, p. 268, see tables 1 and 2]

a	b	a AND b
0	0	0
0	1	0
1	0	0
1	1	1

Table 1. AND gate

a	b	a OR b
0	0	0
0	1	1
1	0	1
1	1	1

Table 2. OR gate

This specification is familiar to both computability theory and mathematical logic. He then presents a third table, this time specifying a processor that takes a pair of inputs of either 5V or 0V, and gives an output of 5V when both inputs equal 5V, or otherwise 0V (see table 3). Is it an AND gate or an OR gate?

Input a	Input b	Output
0 V	0 V	0 V
0 V	5 V	0 V
5 V	0 V	0 V
5 V	5 V	5 V

Table 3. AND gate, or OR gate?

At first glance we are inclined to say that it is an AND gate, but there is in fact no principled way to answer this question without introducing representational content. We have to stipulate that 5 V = 1 and 0 V = 0 in order to determine that it is an AND gate (or vice versa). Sprevak concludes that minimal content is required in order to individuate computational states [1, p. 269].

Sprevak is correct to note that without assigning content it is impossible to non-arbitrarily distinguish 1s from 0s, and AND gates from OR gates, but it does not follow that such an assignment is required in order to individuate computational states. All that is required is that we are able to identify distinctly concatenated digits, and describe the way in which they are processed. Whilst it is convenient for us to give this description in terms of 1s and 0s, we could just as easily have given it in terms of voltages, as in table 3. Brooks captures something of the sense of what I mean when he writes that "[t]he best that can be said [of computation] is that one number is passed from a process to another" [4, p. 144], although in the case of mechanistic computation it is digits rather than numbers that we should speak of.

A voltage level does not *represent* the digit that it is associated with – rather, it *instantiates* that digit. Talk of computation in terms of 1s, 0s, AND gates, and OR gates takes place at a higher level of abstraction than that which is strictly required in order individuate computational states and components. There is perhaps a sense in which computational states come to acquire what Piccinini calls "the *internal* semantics of the computer" [3, p. 214, emphasis in original], but this is very different from how we normally understand representational content. A processor treats all instantiations of the same digit in the same way, and thus might be seen as ascribing representational content to that digit, but this content does not relate in any non-arbitrary way to our external semantics. Piccinini illustrates this by describing a simple instruction in a programming language: "UNTIL P TRUE DO __ ENDUNTIL" [3, p.215]. This causes the mechanism to do __ until the variable P has the value TRUE. However, TRUE here does not strictly speaking mean anything in our external semantics, aside from pointing to the state that P has to be in stop to the mechanism doing __, and this state can be given an entirely physical, non-representational description. We are able to pick out the relevant computational states with this physical description, and hence we are able to individuate them. Any further content that we choose to ascribe to TRUE, or to P, or to __, is entirely irrelevant to the individuation of computational states and components.

To be clear, the claim being made here is that whether a given physical system is computing AND or OR is completely indeterminate from the perspective of computation alone. We can only distinguish the two by attributing representational content to the digits that they are processing, and this process of external attribution is not essential to computation.

5 OBSERVER-RELATIVITY

Sprevak has expressed a concern that my reply to each of these arguments contradicts the judgments of computer science experts (pers. comm.). Whilst experts can be overruled, we need some independent motivation to do so. This motivation can be found in the fact that in each of the above cases representation

introduces an element of observer-relativity into computation. It only makes sense to distinguish the chess-playing computer and the stock-market-predicting computer from the perspective of an external observer. By specification, they are internally identical.[2] The same mechanism could be used for either function, and thus all that differentiates them is the task that we use them for. Similarly, the gate specified by table 3 is computing neither AND nor OR until we attach labels to the digits that it processes.

My claim is that representation, at least in computational systems, can only be specified with reference to an external observer.[3] Thus Sprevak is mistaken in assuming that we can give an account of representation that avoids involving external cognitive agents. If this is true, then we should avoid making use of representational individuation if we want to give an objective account of computation.[4]

Dennett presents a case that can be adapted to help make my point clear. Consider a vending machine that was originally designed to accept American quarters but will also take Panamanian quarter-balboas, which happen to be the same size, shape and weight [7, pp. 290-5]. To the vending machine these two kinds of coins are simply indistinguishable, but to us it makes a considerable difference which kind of coin the machine accepts, as one is worth far less than the other. Imagine that there is an internal state *C* that gets switched on whenever the machine detects either of these types of coin. Any account of what kind of coin this state represents is going to depend on factors external to the machine, such as which country it is in and what kind of coin it most regularly accepts. Thus the representational content of state *C* is determined relative to an external observer, and is not intrinsic to the machine itself.

The same goes for computational states and processes in general – they only represent anything in virtue of an external observer. Of course we can, and often do, give representational specifications to computational states, but this only makes sense relative to an observer, and has nothing to do with the computational process itself. It also picks out nothing intrinsic to the system – the same computation could mean something different to two different observers, as in the case of the chess-playing/stock-market-predicting computer.

If it is the case that representational properties can only be specified with reference to an external observer, then using them as the basis for an objective theory of computation seems doomed to fail. The best we could do is to give a theory of computation that is 'objective' only relative to a given community of observers. It would be 'objective' in the sense that there is an agreed specification of representational content, but not in any universal sense. To give a truly objective theory of computation we would have to reject the received view, and turn to something non-representational, such as the mechanistic account.

6 CONCLUSION

It must be admitted that this is only the beginning of an argument for the claim that representation introduces observer-relativity to computation. Arguments of this sort have tended to be associated with anti-computationalist traditions such as enactivism (e.g. [8]). This association appears to be something of a historical artefact, due to the dominance of the received view in recent decades. My hope is that by defending the mechanistic account it might be possible to free computation from any representational baggage, thus making it available for use by a wider range of explanatory frameworks (both in cognitive science and elsewhere).

There has been a huge amount of literature on representation in recent decades, and so a comprehensive argument for the observer-relativity of representation must be postponed until a later date. My aim here is simply to motivate the idea that a non-representational account of computation would help us to avoid such issues entirely. If I am correct, we must either reject the representational account, and thus the received view, or else accept that computation is observer-relative. As giving an objective account of computation has been a major focus of recent literature, the first option would seem to be the more attractive one.

REFERENCES

[1] M. Sprevak, "Computation, individuation, and the received view of representation" in *Studies in History and Philosophy of Science,* vol. 41, pp. 260-70, 2010.

[2] G. Piccinini, "Computing Mechanisms" in *Philosophy of Science,* vol. 74, pp. 501-26, 2007.

[3] G. Piccinini, "Computation without representation" in *Philosophical Studies,* vol. 137, pp. 205-41, 2008.

[4] R. Brooks, "Intelligence without representation," in *Artificial Intelligence*, vol. 47, pp. 139-59, 1991.

[5] H. Putnam, *Representation and Reality*. Cambridge, MA: MIT Press, 1988.

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

[7] D. Dennett, *The Intentional Stance.* Cambridge, MA: MIT Press, 1987.

[8] D. Hutto & E. Myin, *Radicalizing Enactivism.* Cambridge, MA: MIT Press, 2013.

[2] Aside from their input/output components, which presumably must differ according to whether they play chess or predict the stock market. However, here Sprevak is caught in a dilemma: either these components do not qualify as an essential part of the computing mechanism, or else the two mechanisms are not identical.

[3] In fact I think that the point generalizes beyond computation, but making that additional argument is beyond the scope of this paper.

[4] This claim is related to the familiar accusations of observer-relativity made by Putnam [5] and Searle [6], although I feel that the mechanistic account can limit the problem to representation, and thus avoid the full weight of their criticisms.