Individuation without representation

Individuation Without Representation

Joe Dewhurst

Abstract

Shagrir (2001) and Sprevak (2010) explore the apparent necessity of representation for the individuation of digits (and processors)[1] in computational systems. I will first offer a response to Sprevak's argument that does not mention Shagrir's original formulation, which was more complex. I then extend my initial response to cover Shagrir's argument, thus demonstrating that it is possible to individuate digits in non-representational computing mechanisms. I also consider the implications that the non-representational individuation of digits would have for the broader theory of computing mechanisms.

1. The Received View: No Computation Without Representation

A popular position, which Sprevak (2010) calls the "received view", is that computation requires representation, either for the individuation of states and processes, or in order to give a full account of the causal dynamics of a computational system. I will focus on the former requirement here, as Sprevak considers it to be more fundamental (2010: 261). I will first outline Sprevak's formulation of the requirement, before considering Shagrir's more complex argument.

Sprevak asks that we consider 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*." (Sprevak 2010: 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

[1] As digits and processors are inter-defined, from now on it will be assumed that any reference to digits is also a reference to processors, and vice versa, unless otherwise indicated.

He then asks us to consider a third table that specifies a processor which takes a pair of inputs of either 5V or 0V and returns an output of 5V if both inputs equal 5V, or else 0V (see table 3). Is this processor an AND gate or an OR gate? It seems that without any further information we have no way of answering this question. Whilst our intuition might be that it is an AND gate, because of the convention of associating 0 with low values, there is in fact no reason why it could not be the other way around, with 5V = 0 and 0V = 1 – making it an OR gate. Sprevak concludes that minimal content (i.e. whether 5V represents 1 or 0) is required in order to individuate computational processes (2010: 269).

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?

Shagrir presents a similar case, but this time he asks us to consider a physical system P that consists of a pair of gates that take inputs from two channels ranging from 0V to 10V. One gate outputs 5V-10V if both inputs are over 5V, and 0V-5V otherwise. The other gate outputs 5V-10V if exactly one input is over 5V, and otherwise outputs 0V-5V. (Shagrir 2001: 372-3, see table 4). If we assign 0 to 0V-5V and 1 to 5V-10V, then the first gate is an AND gate and the second gate is an XOR gate (see table 5).[2] Shagrir also notes that we can describe this interpretation as computing addition, by simply treating the outputs of the two gates as binary digits (see table 6). This results in three ways of describing P: physical (i.e. voltages), syntactic (i.e. 0s and 1s), and semantic (i.e. numbers).

Input a	Input b	Gate a output	Gate b output
0-5 V	0-5 V	0-5 V	0-5 V
0-5 V	5-10 V	0-5 V	5-10 V
5-10 V	0-5 V	0-5 V	5-10V
5-10 V	5-10 V	5-10 V	0-5 V

Table 4. Shagrir's voltage gate

Input a	Input b	AND	XOR
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	0

Table 5. 0-5 V = 0, 5-10 V = 1

[2] Alternatively we could flip these assignments, making the first gate an OR gate and the second gate a XNOR gate, but Shagrir does not seem to consider this possibility.

Input a	Input b	AND	XOR	SUM
0	0	0	0	0 + 0 = 00 (0)
0	1	0	1	0 + 1 = 01 (1)
1	0	0	1	1 + 0 = 01 (1)
1	1	1	0	1 + 1 = 10 (2)

Table 6. Interpreted as computing addition for the domain {0,1}

Here the syntactic description is typically taken to be the genuinely computational description of the system, as the semantic description is essentially arbitrary, in the sense that it requires an assignment of content, and the physical description is too demanding, in the sense that it would not allow for multiple realisability. However Shagrir goes on to argue that different syntactic structures may be simultaneously implemented by the same cognitive system, rendering the syntactic structure alone insufficient for computational individuation. He proposes a form of minimal mental content as an additional constraint, in much the same way that Sprevak formulates the representation requirement.

In order to demonstrate that syntax is insufficient Shagrir asks us to image that the gates in our physical system P are in fact "tri-stable", meaning that they are able to distinguish inputs in the 0V-2.5V from those in the 2.5V-5V range. The first kind of gate outputs 5V-10V if it receives 5V-10V from both inputs, 0V-2.5V if it receives 0V-2.5V from both inputs, and otherwise outputs 2.5V-5V. The second kind of gate outputs 5V-10V if it receives exactly one input of 5V-10V, 0V-2.5V if it receives 0V-2.5V from both inputs, and otherwise outputs 2.5V-5V. (Shagrir 2001: 374, see table 7). Note that formulating a table that captures the structure of this system requires an additional 'kind' of input – in effect the system is sensitive to three, rather than two, kinds of digit.

Input a	Input b	Gate a output	Gate b output
0-2.5 V	0-2.5 V	0-2.5 V	0-2.5 V
0-2.5 V	2.5-5 V	2.5-5 V	2.5-5 V
0-2.5 V	5-10 V	2.5-5 V	5-10 V
2.5-5 V	0-2.5 V	2.5-5 V	2.5-5 V
2.5-5V	2.5-5 V	2.5-5 V	2.5-5 V
2.5-5 V	5-10 V	2.5-5 V	5-10 V
5-10V	0-2.5 V	2.5-5 V	5-10 V
5-10 V	2.5-5 V	2.5-5 V	5-10 V
5-10 V	5-10 V	5-10 V	2.5-5 V

Table 7. Shagrir's tri-stable system

If we assign the syntactic symbol 0 to the 0V-2.5V range, and the syntactic symbol 1 to the 2.5V-10V range, then both gates behave as OR gates (see table 8). However, if we return to our previous assignment (0V-5V = 1 and 5V-10V = 0),

then the gates revert to behaving as an AND gate and a XOR gate respectively (see table 9). It seems that the physical structure P simultaneously implements two different syntactical functions (Shagrir 1991: 375). Note that in both cases our truth table now features some redundancy due to the tri-stable system's sensitivities not really being fully captured by our decision to assign only two digits.

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

Table 8. 0-2.5 V = 0, 2.5-10 V = 1

Input a	Input b	AND	XOR
0	0	0	0
0	0	0	0
0	1	0	1
0	0	0	0
0	0	0	0
0	1	0	1
1	0	0	1
1	0	0	1
1	1	1	0

Table 9. 0-5 V = 0, 5-10 V = 1

Shagrir concludes that in order to properly individuate computational processes we require knowledge of the task or function that it carries out, from which we can derive the content of the computation (2001: 382ff). For instance, in the case of the tri-stable system P, we need to know whether it is being used to compute addition or some other function (and thus what the semantic content of its states are) before we can properly determine whether it consists of a pair of OR (or AND) gates or an AND (or OR) gate and a XOR (or XNOR) gate. Thus, representation is required for computational individuation.

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. However, as I will go on to argue, this does not mean that computational processes cannot be individuated without representation – rather, it means that computational processes must be individuated in a way that remains neutral with regard to what logical function they carry out.

2. Computing Mechanisms and Functional Individuation

Piccinini's mechanistic account of computation provides a viable architecture for the non-representational individuation of digits and processes. The account builds on recent work on mechanistic explanation in cognitive science (Piccinini 2007: 501, see e.g. Bechtel 2005; Craver & Bechtel 2006; Craver 2007). As such it aims to decompose the target phenomenon (computation) in terms of the structured interaction of physical components (digits and processors). This will allow for the individuation of digits and processors in purely functional, rather then representational, terms. I will now briefly rehearse the most relevant aspects of the mechanistic account before using it in order to respond to Sprevak and Shagrir's arguments.

For our purposes a computing mechanism is defined as a physical system that carries out systematic transformations on strings of digits. Digits are understood as discrete physical units that can interact with other physical units called processors in order to produce, in a systematic fashion, further discrete physical units (i.e. further strings of digits). For instance, a digit type θ is a component that, when coupled with another digit type θ to produce a string θ-θ, interacts with a processor of type α to produce another digit of type θ. Digit types will be further individuated by a description of how they interact with other processor types, and vice versa for processor types.[3]

I will now outline a response to Sprevak's arguments from the perspective of the mechanistic account of computation. Recall that Sprevak's claim was that minimal representational content is required in order to individuate both digits and processors, as prior to knowing whether 5V represents 0 or 1 we are unable to tell whether the processor in question is an AND gate or an OR gate.

However, consider another processor, as described by table 10. Again, prior to assigning representational content we cannot tell whether this is an AND gate or an OR gate, but note that what we can do is distinguish it from the gate previously described (in table 3), which is its mirror image. This, I contend, is sufficient for computational individuation, and can be achieved with simply a physical description of how the various components of the mechanism function. After individuating the two kinds of processor we are free to assign whatever content we like to the digits in question, but this is strictly irrelevant for computational individuation and is not necessary in order to give a complete description of the computing mechanism.

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

Table 10. AND gate, or OR gate?

It is true that, when considered in isolation, both gates could be used to perform the same logical operation – that is to say, in one system gate 3 might serve as an AND gate, whilst in another system gate 10 could also serve as an

[3] See Piccinini 2007 for a more detailed description of a computing mechanism.

AND gate.[4] In this sense there is perhaps no computational difference between the two gates. However, my point is rather that were the two gates to be placed in the same system, we would easily be able to distinguish between them. In this more restricted sense the physical mechanism alone is sufficient for computational individuation, and no assignment of content is required.

Whilst Shagrir's arguments are somewhat similar to Sprevak's, they require a slightly more complex response. Recall Shagrir's tri-stable gates; depending on our syntactic assignments they can be interpreted either as both performing the OR (or AND) function, or performing the AND (or OR) and XOR (or XNOR) functions respectively. Whilst this true, we do not require any assignment of this kind in order to individuate the gates. It is sufficient for us to note the unique patterns of voltage transformations, as presented in table 11 (this is a repeat of table 7, which can be found on page 3).

Input a	**Input b**	**Gate a output**	**Gate b output**
0-2.5 V	0-2.5 V	0-2.5 V	0-2.5 V
0-2.5 V	2.5-5 V	2.5-5 V	2.5-5 V
0-2.5 V	5-10 V	2.5-5 V	5-10 V
2.5-5 V	0-2.5 V	2.5-5 V	2.5-5 V
2.5-5V	2.5-5 V	2.5-5 V	2.5-5 V
2.5-5 V	5-10 V	2.5-5 V	5-10 V
5-10V	0-2.5 V	2.5-5 V	5-10 V
5-10 V	2.5-5 V	2.5-5 V	5-10 V
5-10 V	5-10 V	5-10 V	2.5-5 V

Table 11. Shagrir's tri-stable system

This is a more complex kind of computing mechanism than that presented in table 10, consisting of two processor types and three digits types, but it is not intrinsically representational (or even syntactic, in any strong sense). This maximal description tells us all that we need to know in order to individuate the components of the computing mechanism, and is sufficient to give a precise account of any computation that it might perform. The two descriptions that Shagrir gives in tables 8 and 9, on the other hand, can only be achieved by making a decision about how to group the three digit types in table 7, so as to end up with a system that only appears to have two digits. The decision that we come to about how to group the digits will of course depend on the syntactic and semantic assignments that we make, but these are strictly irrelevant to the functioning of the system *qua* computation – or so says the mechanistic account.

A further complication arises if we compare Shagrir's tri-stable system (table 11) with a bi-stable system consisting of two processors, each equating to one of Shagrir's gates but sensitive only to 0-5V and 5-10V.[5] Both could be used for computing addition, in which case we would interpret 0-5V as representing 0, and 5-10V as representing 1. These systems appear to be performing the same

[4] Mark Sprevak brought this point to my attention.

[5] An anonymous reviewer brought this point to my attention, along with the subsequent point about no two physical systems being strictly equivalent. The case presented here is a version of one that they suggested.

algorithm, which would traditionally be sufficient for computational equivalence. However, they are structurally distinct, as the tri-stable system is sensitive to difference in voltages that the bi-stable system is not. According to my interpretation of the mechanistic account this means that they cannot be computationally equivalent, as one is a mechanism consisting of *three* digits and two processors, whilst the other consists only of *two* digits and two processors. The fact that we interpret them as performing the same mathematical operation is, if you accept my argument, strictly irrelevant to their computational individuation.

This would be problematic if one thought that it was important to see these systems as computationally equivalent, for example in order to be able to theorize about their equivalent algorithmic status as adders. Something along these lines is the position traditionally held with regard to systems such as these. On the contrary, though, what I think this case actually demonstrates is the importance of keeping algorithmic and computational equivalence (or lack thereof) distinct. It is indeed an interesting feature of these two systems that we can use both of them to compute the same mathematical operation. It is also interesting that we can do this despite them both possessing distinct computational structures. By conflating computational and algorithmic equivalence we risk missing this second interesting feature, giving us reason to reject representational individuation.

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. One option would be to appeal, as Piccinini does (2007: 507), to the distinctions drawn by the relevant experts, which might allow us to say that, for example, voltage levels but not temperature are relevant. Another option would be to simply bite the bullet and accept that no two physical computing mechanisms are equivalent, at risk of inviting a potential *reduction ad absurdum.*

Whichever way we decide to go on this issue, it is worth noting that the problem is shared by mechanistic accounts more generally, not just the mechanistic account of computation (see e.g. Craver 2009 for a discussion). Consider a mechanistic account of the cardiovascular system. Such an account might invoke components such as veins, arteries, and red blood cells, and go on to posit equivalence between systems that have these components in common. Of course no two cardiovascular systems will be equivalent in all aspects of physical structure, but experts in cardiovascular science will specify which aspects are relevant, and this might be the best that we can do. Ultimately there is going to be a degree of observer-relative arbitrariness to any scientific practice, and this does not seem to be any worse in the case of computer science than in other scientific disciplines. The target phenomenon, in this case something like the structured transformation of digits, will constrain which physical properties are to be considered relevant – i.e., those that are detectable by the processing component(s) of the computing mechanism in question. In this sense the fortunes of the mechanistic account of computation are hostage to the

fortunes of mechanistic accounts more generally, but a full defence of such accounts is beyond the scope of this paper.

Piccinini (2008; 2015: chapter 3) puts forward a similar response to that which I have outlined above, but he concedes that multiple computations are implemented by Shagrir's system. Which of the implemented computations is most explanatory, he continues, will depend on the context within which they take place. I agree that the context of implementation is important for determining which *logical operation* is of explanatory interest, but I want to deny that this context relativity extends to the individuation of computational processes, which are fully captured by the physical structure of the computing mechanisms. This has some fairly counterintuitive implications, which I will consider in the next couple of sections.

3. Against Computational Externalism

Shagrir claims that the representation condition provides partial support for what he calls "computational externalism", namely, the thesis that "Computational theories of cognition make essential reference to features in the individuals environment" (2001: 392). In presenting this claim he equates computational structure with cognitive processes, which I believe to be a false equivalence. It is therefore important to respond to his claim here, and to clarify the relationship between computational individuation and cognitive scientific explanation.

Shagrir asks us to consider a case where two computational systems come to possess a distinct syntactic structure, in virtue of the distinct tasks that they have been designed to solve (2001: 397). This is intended to demonstrate a case where external factors determine computational individuation. His description of this case is brief, and as I see it there are two ways of interpreting it. Either the two systems are physically distinct, in which case all he has demonstrated is the almost certainly trivial point that physically distinct computational systems are also computationally distinct. Or the two systems are physically identical, in which case we need to give a more thorough account of what it means for them to be syntactically distinct. I will focus on the second interpretation, as it is the more charitable.

If the account given in the previous section is correct, then the received view is false, and the equivalence that Shagrir draws between computational and syntactic identity is invalid. This means that for syntactical structure to vary whilst physical structure is held fixed we need something like Shagrir's tri-stable system, which is open to at least two distinct syntactic interpretations. As we have already seen, however, this kind of case does not pose any threat to the non-representational individuation of computational systems. Shagrir is correct in noting that syntactic (or representational) content is at least partially determined by external factors, but this kind of externalism need not carry over to computational individuation, which can be accomplished entirely in terms of intrinsic physical structure.

Shagrir's argument for computational externalism is undermined by the distinction that the mechanistic account draws between computational and syntactic/representational structure. This does not imply that externalism about *cognition* is necessarily false, but only that if externalism about cognition is true

then it must be true in virtue of something other than the computational aspect of cognition. If computational states and processes can be individuated without representational content, then we should be able to give a completely internalist account of computation. Cognition, on the other hand, might require more than computation, leaving room for some form of externalism with regard to content.

The advantage of this approach is that provides a principled way of separating what we might call 'mere' computation (i.e. computation without representation) from full-blown cognition. If we allow for representational content to determine the individuation of computational states then we risk getting trapped in a kind of "vicious" circularity, where computational structure determines content at the same time as content determines computational structure (see Piccinini 2004a, 2004b). By giving a distinctive, non-representational description of computation, the mechanistic account provides room for computational structure to contribute to the determination of representational content without inviting any kind of circularity or regress. Computational individuation remains important because we want to be able to identify the computational structures that instantiate cognitive systems, but the determination of cognitive content turns out to require the invocation of additional, non-computational factors, some of which may even be external to the system in question.

4. Implications For The Mechanistic Account

I have demonstrated that it is possible for the mechanistic account to individuate digits and processors without invoking the notion of representation. However, this position does not come without costs – in particular, it leads to some fairly counterintuitive implications for our understanding of computation. Here I will focus on just one of those implications – computation, in the physical/mechanistic sense, becomes distinct from the abstract logical operations that it is typically thought to implement. It also turns out that so long as we require an account of mental representation from our theories of cognition, computation alone will be insufficient for the constitution of cognitive systems.

If, as I have argued, computational processes can be individuated without invoking any representational content, however minimal, then the logical status of those processes, which requires the attribution of minimal content ('true' or 'false'), will no longer be an intrinsic feature of computation. Recall that both Sprevak and Shagrir's arguments rested on the logical indeterminacy of the physical systems captured by tables 3, 4, 7, and 10. Without knowing which binary value to assign to each voltage level, it becomes impossible to determine what kind of logic gate these systems implement. My response was to accept this indeterminacy, but point to the fact that each system is still distinguishable from its inverse, and thus we are able to individuate them. To make this absolutely clear, imagine a physical system consisting of two processing components captured by tables 12 and 13 (these are identical to tables 3 and 10, found respectively on pages 2 and 5) connected in parallel. They each take the same pair of digits as inputs, and together they produce a pair of digits as output (one from each processor). If, for instance we gave them 0 V + 5 V as input, they would produce 0 V (from gate 3) + 5 V (from gate 10) as output.

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 12. 'Gate 3'

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

Table 13. 'Gate 10'

It should be apparent that this system as a whole does something comparable to Shagrir's first voltage gate (see table 4 on page 2). If we were to assign binary numerical values to the voltage levels then the system could be interpreted as performing addition for the domain {0, 1}. Which voltage level is 0, and which is 1? This, perhaps ironically considering Sprevak's formulation of this problem, depends on which processor we label as an AND gate, and which we label as an OR gate. If 3 is an AND gate and 10 is an OR gate, then 5 V = 1 and 0 V = 0, but if 3 is an OR gate and 10 is an AND gate, then 5 V = 0 and 0 V = 1. This does not mean that we cannot individuate the components of the system (tables 3 and 10 do just that), but rather that our individuation of this system will not make any reference to the logical values of the digits or processor. Computational individuation, it seems, is in some sense prior to logical individuation.

To many this will seem extremely counterintuitive. What could be more fundamental to our understanding of computation than logic? Historically this may have been the case, but I think it is clear that the logical status of the system described above is indeterminate, whilst its computational status, in the mechanistic sense, is not. What this allows for is the grounding of our logical apparatus in a non-logical physical system, in much the same way as I described for representation at the end of the previous section. The benefit of this is a degree of objectivity it what might otherwise seem like a very subjective attribution of representational (or logical) content, but the cost is biting the counterintuitive bullet of making logic (in physical systems) strictly dependent upon pre-existing computational structure.

Another potential concern is that a totally non-representational account of computation might seem to provide a poor basis for a theory of cognition, which is frequently assumed to be a paradigmatically representational enterprise. Shagrir's argument for computational externalism, discussed in the previous section, highlights this concern. The arguments for externalism that he dismisses fail primarily because they are concerned with the semantic content of mental states, not the formal structure of a computational system. The latter, I want to claim, is determined entirely by internal, mechanistic factors, whilst the

former might well require reference to the external world. In fact, depending on what theory of mental content we adopt, it seems that some form of cognitive externalism will be quite likely to follow. I am not concerned with how that debate plays out – provided that computational individuation is kept entirely separate and prior to the determination of mental content, it will not matter for my purposes whether or not we think cognitive externalism is true.

There also remains the lingering threat of pancomputationalism. By reducing computation to a mere physical process, one might worry that it becomes trivially easy to identify computation in all kinds of unseemly places, such as the infamous wall that implements WordStar (Searle 1992: 208-9). However I think this is less of a concern than it might first appear – the only physical systems that will compute are those where distinct computational components can be identified. Searle's WordStar wall does not contain anything resembling strings of digits or processors, and treating it as a computational system does not gain us any explanatory purchase. In contrast, even a minimal computational system such as that captured by table 3 is amenable to mechanistic explanation. We know what a digit looks like (5 V or 0 V), and we know what will happen when we feed it certain strings of digits (0 V + 5 V will be converted to 0 V, for example). What we do not know is whether this computational system is implementing an AND gate or an OR gate, but my main message in this paper has been that this should not surprise us. By itself a voltage level does not mean anything, and if as a result of this it turns out that the logical structure of computational systems is context relative, then so be it.

Acknowledgements

Oron Shagrir prompted me to write this paper, and Mark Sprevak gave invaluable comments and feedback on several earlier drafts. An anonymous referee gave a very detailed review, which led to extensive revisions that improved the paper greatly. Any remaining errors or inaccuracies are entirely my own.

References

- Bechtel, W. 2005. "Mental Mechanisms: What are the operations?" *Proceedings of the 27th annual meeting of the Cognitive Science Society*: 208-13.
- Craver, C. & Bechtel, W. 2006. "Mechanism." In Sarkar & Pfeifer (eds.), *Philosophy of Science: an encyclopedia*. New York: Routledge.
- Craver, C. 2007. *Explaining the Brain*. Oxford: OUP.
- Craver, C. 2009. "Mechanisms and natural kinds." *Philosophical Psychology* 22/5: 575-94.
- Piccinini, G. 2004a. "Functionalism, computationalism, and mental contents." *Canadian Journal of Philosophy* 34/3: 375-410
- Piccinini, G. 2004b. "Functionalism, computationalism, and mental states." *Studies in History and Philosophy of Science* 35/4: 811-33.
- Piccinini, G. 2007. "Computation mechanisms." *Philosophy of Science* 74/4: 501-26.

- Piccinini, G. 2008. "Computation Without Representation." *Philosophical Studies* 137(2): 205-241.
- Piccinini, G. 2015. *Physical Computation: A Mechanistic Account.* Oxford: OUP.
- Searle, J. 1992. *The Rediscovery of Mind.* MIT Press.
- Shagrir, O. 2001. "Content, Computation and Externalism." *Mind* 110/438: 369-400.
- Sprevak, M. 2010. "Computation, individuation, and the received view on representations." *Studies in History and Philosophy of Science* 41: 260-70.