Philosophers' Imprint

volume 18, no. 24
december 2018

How to Explain Miscomputation

Chris Tucker
College of William & Mary

Introduction

Just as a theory of representation is deficient if it can't explain how misrepresentation is possible, a theory of computation is deficient if it can't explain how miscomputation is possible. You might expect, then, that philosophers of computation have well-worked-out theories of miscomputation. But you'd be wrong. They have generally ignored miscomputation.[1]

My primary goal in this paper is to clarify both what miscomputation is and what needs to be accomplished in order to adequately explain it. Miscomputation is a special kind of malfunction. If the battery breaks, a system may fail to compute what it is supposed to compute. But it's not miscomputing, because it's not computing at all. Just as something doesn't misrepresent unless it represents, something doesn't miscompute unless it computes. To miscompute is to compute in a way that violates a computational norm. Consequently, an adequate account of miscomputation requires an account of what the system is computing when the system is violating the relevant computational norms.

A secondary goal is to defend an *individualist* approach to miscomputation. The advantage of this account is that it provides a simple and straightforward explanation of miscomputation. Piccinini contends that, by appealing to teleological functions, his externalist account also enjoys this advantage. It does not. Not yet, anyway. Following just about every discussion of functional individuation, Piccinini focuses on how to individuate *proper* function.[2] I focus instead on how to individuate *actual* function. This distinctive focus reveals that Piccinini-inspired approaches struggle to account for the computational structure of *mal*functioning systems.[3]

1. This apparent fact is bemoaned by Dewhurst (2014), Fresco and Primiero (2013: 254), and Piccinini (2015: 14, 48).
2. Allen's *SEP* entry on "Teleological Notions in Biology" does not even raise the question of how to individuate a biological system's behavior when the behavior fails to fulfill its (proper or teleological) function.
3. Thanks to Gualtiero Piccinini for pointing out that my approach was distinctive.

The strict individualist account that I favor may allow solar systems to compute. For those who are bothered by this drawback, I briefly introduce *quasi-individualism* at the end of the paper. It seems to get the advantages of my individualist approach to explaining miscomputation while nonetheless avoiding its drawback.

I assume that a mechanistic (functional) theory of computational individuation is true and that content is not needed to individuate computational structure. Proponents of this framework tend to be externalists, so in §1, I explain why we should take an individualist version of the mechanistic theory seriously. In §2, I show that this individualist, mechanistic theory easily accounts for miscomputation. In §3, I criticize an externalist approach to explaining miscomputation inspired by Piccinini's work. In §4, I briefly introduce the quasi-individualist theory that may get the advantages of both my account and Piccinini's without any of the disadvantages.

1. Computational Structures and Functional Structures

In this section, I explain why an individualist mechanistic theory is worth taking seriously. In §1.1, I'll rehearse a mostly familiar rationale for the mechanistic theory, while highlighting its neutrality between externalism and individualism. In §1.2, I introduce an individualist mechanistic theory that is modeled on Piccinini's externalist version. The near identity between the two views will make it easier to understand, in §1.3, a certain advantage of going externalist and why it is a small one.

1.1. Mechanistic Theories of Computational Structure

I focus on digital computation. Oversimplifying, a digital computing system is an input-output device in which the relevant inputs and outputs are strings of digits. A digit is a state of the system. In simple systems, digits are often just electrical charges, where charges of different voltages can count as different digits. A string of digits is an ordered list of digits. For example, a string of digits in a system might be a series of electrical charges. A system computes when the formal properties of the input string "lead" the system to produce a certain output string, e.g., a different series of electrical charges. In the abstract, a **computational structure** is a complete mapping from the possible input strings to the possible output strings. A system **has** or **implements** a certain computational structure when the structure, or mapping, is a correct description of the system's (actual and counterfactual) behavior. That much is relatively uncontroversial.

A simple mapping account of computation imposes no restrictions on which mappings from inputs to outputs capture the computational structure of the system. Such views trivialize computation by being committed to **unlimited pancomputationalism**, i.e., the idea that *every* physical system computes *every* computation. To do better, we must impose restrictions on which mappings capture the computational structure of a system.

The first restriction is to endorse what we can call a **functional theory**. Such views hold that a system's computational structure is determined by (a specific kind of) dispositional or functional structure.[4] On this view, to say that a system implements a certain computational structure is not merely to describe the system: it is to explain, at a certain level of abstraction, why the system is doing what it's doing (or would do were it given a certain input). The relevant type of explanation is dispositional explanation. We explain what the salt is doing while submerged, in part, by pointing out that it is soluble, that it has a certain dispositional structure. When we explain a system's behavior by appealing to a dispositional structure that counts as computational, we provide a computational explanation of the system's behavior.

The functionalist theory — independently of any commitment to individualism or externalism — grounds a response to familiar worries about individuating digits. We could treat any voltage up to five volts as a single digit. Alternatively, we could treat any voltage less than 2.5v as single digit and any voltage from 2.5 to 5v as a distinct digit. The

4. I clarify the relation between dispositions and functions in §2.3.

functionalist theory claims that digits are individuated according to their functional significance in the system. If a system's outputs aren't differentially sensitive to input voltages of ≤5v, then the computational structure of a system treats those voltages as a single digit rather than two. The functionalist picture, so construed, determines neither a unique binary syntax (e.g., assignment of 0s and 1s to voltage ranges) nor a unique logical function (e.g., and-gate vs or-gate). I take this to be a feature, not a bug. A natural corollary of the functionalist theory is that computational individuation is explanatorily prior to both the semantics and the binary syntax of the system.[5] (We can still follow the convention of stating computations in terms of 0s and 1s, as long as we keep in mind that this way of talking is partly mere convention.[6])

While the simple functionalist view is an improvement over the simple mapping account, it is still too liberal. Since systems implement only those computations that track their functional structure, it avoids unlimited pancomputationalism (cf. Dewhurst 2018: 115). Yet it doesn't avoid **limited pancomputationalism**, the idea that *everything* computes *at least one* computation (cf. Chalmers 2011: 331). It also seems committed to the idea that digestive processes implement some computation or another (cf. 332).

We can make a little more progress by endorsing a **mechanistic (functionalist) theory** of computation. To be a mechanism is, among other things, to have parts whose integrated operation explains its overall behavior. Mechanistic theories hold that only mechanisms compute and that the computationally relevant functional structure is restricted to the functional structure of the system that is grounded in the integration of its parts.[7] All mechanistic theories, therefore, deny that non-mechanisms compute. This denial provides even the crudest mechanistic theories with some protection against limited pancomputationalism. To be a mechanism requires both parts and integration among those parts. Mereological simples and mere mereological composites lack parts and integration, respectively.[8] Thus, they don't compute.

One way to make further progress against limited pancomputationalism is to provide conditions on which sorts of integration are necessary for the existence of a mechanism. Yet I will assume that it takes very little integration to count as a mechanism. For example, I assume that solar systems (and rocks and many other objects that intuitively do not compute) *do* have the narrowly individuated integration necessary to count as mechanisms. This assumption makes it harder to show that individualism is worth taking seriously.

Another way to get further distance from limited pancomputationalism is not to impose further restrictions on what it takes to have mechanistic structure, but to impose further restrictions on which mechanistic structures are computational. Digestive organs are mechanisms, but digestive processes intuitively do not implement computation. Following Piccinini, I require that computationally relevant functional structure be **medium-independent**. To count as a computing system, a system's behavior must be explicable at a level of abstraction that makes no reference to the media in which the behavior is carried out.

Medium-independence is stronger than multiple realizability. There is more than one way to realize *removing corks from wine bottles*, but this behavior is necessarily performed on certain kind of media,

5. See Dewhurst 2018 (especially 110–1) for clarification and further defense of this functionalist response to Shagrir's and Sprevak's individuation worries. My one caveat is that Dewhurst incorrectly (or misleadingly) claims that computational individuation occurs at the *physical* level. Coelho Mollo (2018, sec 7) correctly notes that Dewhurt's response can be improved by holding that computational individuation occurs at the *functional* level.

6. Relatedly, many find it useful to state computations using the normal conventions even though they explicitly acknowledge that it is computationally irrelevant whether we, e.g., assign '0' or '1' to the lower of two functionally relevant voltage ranges (Fresco 2015: 1050–1; Piccinini 2015: 142; and Shagrir 2001: 373). For an alternative point of view, see Sprevak 2010: 268–9.

7. See Coelho Mollo (2018) for the ways in which mechanistic theories of computation are related to functional and mechanistic explanation more generally.

8. I take it that the mereological composite of my nose and computer doesn't compute simply because it has a part that computes. It is commonly assumed, for example, that the solar system doesn't compute even though it contains computers.

namely wine bottles and corks. This behavior is multiply realizable but not medium-independent (Piccinini 2015: 122–3). Medium- independence imposes a significant constraint on which functional structures are computational. The behavior of the digestive system is, for example, "quintessentially medium-*dependent*" (147). Its processes are defined in terms of "specific chemical changes to specific families of molecules" (147).

Computational structures, then, are medium-independent functional structures. *These* structures individuate digits. What individuates a digit is *not* that it plays some functional role in the system at some level of abstraction, as long as there is some (perhaps distinct) level of abstraction in which the system has medium-independent functional structure. Rather:

> **Medium-Independent Individuation:** to be a digit is to play a certain kind of role *in* the medium-independent functional structure of the system, and distinct digits are further typed according to their more specific roles in that structure (cf. Piccinini 2015: 122; cf. 127–8).

As we'll see in §3.2, respecting Medium-Independent Individuation in the context of malfunction is easier said than done.

All progress so far is neutral between individualism and externalism. We've seen that a specific kind of mechanistic theory of computation — one that appeals to medium-independent, mechanistic functional structure — avoids three vices: trivializing computation, unlimited pancomputationalism, and allowing digestive processes to be computational. Technically, the account also avoids limited pancomputationalism insofar as it denies that non-mechanisms compute. But this may seem a mere technicality. We still lack the resources to deny that solar systems compute.

1.2. An Individualist Mechanistic Theory

The most promising mechanistic theories will impose further restrictions beyond those mentioned in the previous sub-section — both restrictions on which sorts of functional structure are computational (whether there is computation at all) and restrictions on which differences in functional structure make a computational difference (which computation is implemented). Let the **definitive list** be the complete list specifying which properties are necessary and sufficient for a dispositional structure to be computational and which properties further individuate a system's computational structure. The definitive list, then, is the correct and complete account of which functional structures count as computational and which differences between functional structures make a computational difference. I'll pretend that Piccinini has given us the definitive list, but even he admits that his list is incomplete (2015: 120).[9]

My individualist account is identical to that of Piccinini (2007; 2008; 2015), save two differences. This near identity means that my account inherits the incompleteness of Piccinini's. The first difference concerns the scope of computational individuation. He claims that it is individuated *widely*: the environment of a system, including any larger system of which it is a part, can make a difference to what computational structure the system has. I claim it is individuated *narrowly*: the environment (broadly construed) cannot affect the computational individuation of a system. He's a computational externalist, and I'm an individualist. The second difference concerns whether the computationally relevant functional structure involves normatively loaded teleology, i.e., whether the computationally relevant functional structure determines what the system *should* be doing. He says yes; I say no.

We won't be able to fully appreciate these differences, especially the second one, until the end of §2. In the meantime, I'll assume that you have at least a crude idea of what the differences amount to. In the rest of this sub-section, I'll consider a worry about my individualist account that arises because of the first difference. In the next sub-section, I'll consider a worry about the individualist theory that arises because of the second difference.

9. For his working list, see, e.g., 2007: 508–14, 2015b: 120–34.

Suppose that S is a computing component of some larger system S*. Piccinini assumes **Digital Perseverance**: necessarily, feature F counts as a distinct digit for S only if F counts as a distinct digit for S*. The computational structure of the whole constrains how digits are individuated for the part (2008: 229, 2015: 41; cf. Bontly 1998: 570 and Segal 1991: 492–3). Let S be Shagrir's (2001) tri-stable system, which is differentially sensitive to three different voltage ranges: (i) ≤2.5v, (ii) between 2.5v and 5v, and (iii) >5v. Suppose S* computes over two digits, ≤5v (0) and >5v (1). Digital Perseverance says that S can't operate over three digits, <2.5v (0), 2.5v up to 5v (½), and ≤5v (1). If Digital Perseverance is true, then so is computational externalism. For, given Digital Perseverance, the computational structure of a part can't supervene on *its* physical structure, though it might supervene on the physical structure of the whole mechanism of which it is a part (cf. Segal 1991: 492–3).

I reject Digital Perseverance. Where Piccinini sees the computational structure of the larger mechanism imposing constraints on the computational structure of the component part, I see computational significance getting lost in composition. First, it is uncontroversial that a complete and correct computational description of a whole device and each of its parts is compatible with the parts performing different, usually simpler,[10] computations than the whole mechanism (cf. Egan 1995: 192). If the part and whole can perform different computations, why can't the part perform computations on more digits than the whole? Why *must* the whole mechanism care about everything each of its parts cares about? Without an answer to these questions, there is no reason to deny that parts can compute over more digits than their wholes.

Second, it's possible to have a computing device, such that one of its computing parts is differentially sensitive to three voltage ranges when the device as a whole is differentially sensitive only to two voltage ranges.[11] If you think that computational structures track functional significance, as Piccinini and I do, then you have some reason to treat the larger device as computing over fewer digits than one of its parts.

My concerns about Digital Perseverance are hardly decisive, but my goal is not to prove that my individualistic mechanistic theory is true. The goal is just to show that the theory is worth taking seriously enough to explore how, if true, it might contribute to an adequate explanation of miscomputation.

1.3. Solar Systems: A Comparison

Piccinini's externalist mechanist account does not allow solar systems to compute. That's because solar systems "are not collections of components functionally organized to fulfill specific teleological functions" (145).[12] There is no teleological function of solar systems, because there is nothing they *should* be doing, and there is certainly no function that they *should* be computing. I allow there to be computation without teleological function, and Piccinini doesn't.[13] He is, therefore, better positioned than I am to deny that solar systems compute. That's an advantage of going externalist. But it's a small one.

Piccinini allows that something can have a teleological function to compute function F because it is used to compute F or a designer intends it to compute F (2015: 148–9). But there are limits. Usage or

10. Parts tend to perform simpler computations, because systems are generally constructed so that their computing operations are somewhat efficient. It's possible for the parts to perform more complex computations than the whole.

11. You'll save yourself some time if you take my word for it. For the most scrupulous readers, I describe such a device. Let S3 be the composition of S1 and S2. Suppose S1 is differentially sensitive to three voltage ranges: <2.5v, 2.5v–5v, and >5v. Let S1's outputs be S2's inputs, where S2 is only bi-stable, and thus only differentially sensitive to two voltage ranges, 0–5v and >5v. Whenever S2's input is 0–5v, it outputs 0–5v. Whenever its input is >5v, it outputs >5v. In such a case, a system S3, which is solely composed of S1 and S2, will be differentially sensitive to only two ranges, 0–5v and >5v. Assuming that Digital Perseverance is false, S3 is an and-gate.

12. Piccinini also points out that arbitrary inputs into a solar system do not play the relevant functional role within a system to count as digits; however, this additional point is neutral between individualism and externalism.

13. My individualist account appeals to functional structure that is *not* teleological. I clarify this sort of structure in §2.3.

intentions can bestow teleological function on S only if S (or at least properly functioning members of S's kind) *can* compute the functions it is used or intended to compute (Piccinini 2015: 149, (iii)). In other words, a system's narrowly individuated functional structure imposes constraints on which teleological functions it has. It can't be a teleological function of a marvelously colored rock that it run *Minecraft*; however, since the rock can serve as a paperweight, it presumably can acquire the teleological function of paperweights.

Let us say that system S is **eligible** to compute function *F* iff S has narrowly individuated functional structure compatible with computing *F*. In other words, S is eligible to compute F iff there is some possible environment in which S computes F.

Anything that is eligible to compute on my account is also eligible to compute on Piccinini's. Assume that a system is eligible to compute on my account. Then it has medium-independent, narrowly individuated functional structure that satisfies the definitive list. This is sufficient for computation on my account (my individualist view entails that systems compute every function that they are eligible to compute). For Piccinini, this medium-independent, narrowly individuated functional structure doesn't suffice for computation. Yet it does suffice to be eligible for computation. In such a case, to get computation, on his view, the environment need only supply the system with a telos to compute a function that its medium-independent functional structure can compute. And such a telos can be provided by usage or intention.

I'm committed to saying that solar systems compute whether anyone bothers to use or intend them to compute. Piccinini denies that solar systems compute, but he's committed to saying that they would compute if anyone (e.g., a god or supersmart alien) bothered to intend or use them to compute. This difference does not strike me as a sound basis for preferring one account of computational individuation over another. At any rate, any advantage here is a small one.

My account seems to have an overcounting problem. In other words, my account sometimes says that a system computes (a given function) when, intuitively, the system doesn't do so. Yet not all overcounting problems are equally bad. A theory has a **qualitative overcounting problem** iff: the theory claims that a system computes function F when the system is not even eligible to compute F. A theory has a **(merely) quantitative overcounting problem** iff: the theory claims that a system is computing function F when it isn't, where the system is at least eligible to compute F. Qualitative overcounting problems are the more serious problem. They are category mistakes. It is a category mistake to call a rock a person. The rock's internal structure prevents it from being a person, and so there is no possible environment in which the rock is a person. Likewise, it is a category mistake to say that the rock implements the computations necessary to run *Minecraft*. Its internal structure prevents it from implementing those computations, and so there is no possible environment in which it does.

A primary reason why my overcounting problems are, at most, a small advantage for Piccinini is that there is no way for Piccinini to claim that I have a qualitative overcounting problem that he doesn't have.[14] For my theory never allows a system to be eligible that isn't also eligible under Piccinini's theory. My alleged problems are merely quantitative. In §3.1, we'll see that Piccinini's overcounting problems are worse: he has *qualitative* overcounting problems that arise in the context of malfunction.

2. Miscomputing Individualistically

My individualist, mechanistic account of computational individualism isn't obviously correct, but it is worth taking seriously. We'll get more reason to take it seriously when we see how easy it makes explaining miscomputation.

14. I'm assuming, remember, that something in the neighborhood of Piccinini's mechanistic theory is true. I haven't ruled out that a very different theory of computation would enjoy an advantage over mine with respect to qualitative overcounting.

2.1 Miscomputation

We can better understand *miscomputation* by considering some analogies. To misbehave is to behave in a way that violates some relevant norm (e.g., the norms of rationality, morality, or etiquette). To miscommunicate is to communicate in a way that violates some relevant norm (what is communicated is not what was *intended* to be communicated). To misrepresent is to represent in a way that violates some relevant norm (e.g., truth). You get the picture. To **miscompute** is to compute in a way that violates some relevant norm — more specifically, a norm for what the system *should* be computing.

A miscomputation, so understood, is a special kind of malfunction. It is special in two ways. First, while malfunctions require normativity at some level of description, miscomputation involves normativity, more specifically, at the computational level. Where there is no computational behavior that the system should be performing, there is no miscomputation. Second, not all computational failures count as miscomputations. Some such failures are merely mechanical. If the battery breaks, the computing system won't compute anything at all. And if it doesn't compute anything at all, it's not miscomputing, just as a diagram doesn't misrepresent unless it represents. To miscompute is to compute.

In this paper, I focus on a specific kind of miscomputation, the kind in which a system implements one computation when it should have implemented a distinct computation.[15] The existence of such miscomputation should be taken seriously. There are cases of misbehavior in which we can identify what the child is doing (throwing his broccoli across the room) and what he should be doing (eating the broccoli). There are cases of misrepresentation in which we can identify what an experience represents (one horizontal line is longer than another) and what counts as correct representation (both horizontal lines are the same length). There are cases of miscommunication in which we can identify what was actually communicated (I don't like your gift) and what was intended to be communicated (I like the gift so much I want to save it for a special occasion).

15. A different kind of miscomputation might occur when a system performs a single computation when it should have performed multiple computations in parallel.

An adequate account of (the relevant sort of) miscomputation involves at least three components: an account of computational behavior (what computation, if any, a system *is* performing); an account of computational norms (what computation[s] the system *should be* performing); and an explanation of how these two accounts together make it possible for a system to compute in a way it should not be computing. The latter explanation may be as trivial as pointing out that, in circumstances C, the account of computational behavior entails that the system computes f_1 when the account of computational norms says that what should be computed is a distinct function f_2.

2.2. Miscomputation Explained

To make the discussion manageable, I make two simplifying assumptions. First, I assume that a system always manifests its dispositions (and so computational structure) when triggered by the relevant input conditions. This allows us to ignore various complications, such as the possibility of masking, performance error, etc. Witches and protective Styrofoam can mask a vase's fragility so that the vase won't break when dropped. We set aside such possibilities. We assume that, when dropped, a vase will manifest its fragility by breaking. A computing system likewise performs the computations that manifest its computational structure. Second, I assume that all computation is deductive or non-probabilistic. When such a computing system receives a computational input and manifests its computational structure, it is guaranteed to produce a specific computational output. These two assumptions are harmless. Any adequate account of miscomputation will allow a computing system to miscompute when it manifests a deductive computational structure. These assumptions simplify our discussion by letting us assume that any difference in computational behavior must be explained by a difference in computational structure.

Recall from §1 that, in the abstract, a computational structure is a complete mapping of computational inputs to computational outputs. A physical system has a certain computational structure iff the structure/mapping counts as a correct description of the system's actual and counterfactual behavior. The computational structure of a system tells us what the system would do were it to receive a given computational input. Perhaps when given string 0,1 as an input, it outputs 1. When you know the actual inputs to the system (and you assume that a deductive computational structure is manifested), the computational structure tells you the actual computational behavior of the system. In other words, computational structure + computational inputs = computational behavior.

Any account of computational behavior will have this same basic structure. The main difference between rival accounts of computational behavior will be their respective accounts of computational structure.[16] When circumstances remain fixed, a difference in computational behavior requires a difference in computational structure.

My account of a system's computational behavior begins, naturally enough, with my account of computational structures. Recall that, on my view, a system's computational structure is determined by its narrowly individuated, medium-independent functional structure (that satisfies the definitive list). To determine what computations are being performed by the system, just plug in the computational inputs (i.e., those states of the system that play the relevant kind of functional role in the behavior of the system).

I deny, however, that narrowly individuated, medium-independent functional structure determines the computational *norms* for the system. The norms that guide a system's computational behavior are given, at least in part, by something external to the system itself, e.g., the evolutionary history of the system, the intentions of a designer, the role that system plays in some larger system, etc. To account for miscomputations, just pick your favorite account of the normativity associated with functional roles. Any of the standard accounts will do,[17] and so will Piccinini's (2015, ch 6). Any of them are sufficient for my purposes, because they individuate norms widely. Since computational behavior is individuated narrowly and computational norms are individuated widely, it's easy to see that a system can compute a function that it isn't supposed to compute.

Suppose, for illustration, that the normativity for a manufactured system can be supplied by the intentions of the designer. If so, then miscomputations can arise because of design error (cf. Piccinini 2015: 149). I might intend for system S to compute function f_1 but mistakenly construct it so that it computes function f_2 instead. To compute f_1, perhaps the system needs to be differentially sensitive to three voltage ranges when its current construction makes it differentially sensitive to only two. In such a case, S would be miscomputing.

Or suppose that S is a computing component of some larger biological system S*. In order for S to make its essential contribution to the biological fitness of S* (or whatever determines S's teleological function), it needs to compute function f_3. But a component of S is damaged (S has a brain lesion, perhaps), so it computes f_4 instead. S is miscomputing. f_4 is the computation that actually explains the behavior of S, when it should have behaved so as to be correctly explained by f_3. Again, given my account of computational structure and any standard account of computational norms, S would be miscomputing.[18]

16. Rival accounts of computation can disagree about what counts as the computational input to the system, i.e., what counts as a digit, but these differences will reduce to disagreements over the computational structure of the system

17. See Allen (2003) for a survey of the standard accounts for biological organisms.

18. For simplicity, I assume that only one set of computational norms will apply to a given device. If it is possible for a device to be subject to conflicting computational norms (e.g., I intend that a device compute a certain function and you intend it to compute a distinct function), then a device may miscompute relative to one set of norms without miscomputing relative to all norms that apply to it.

2.3. Two Senses of Functional Structure

To better understand how my account of miscomputation works, we need to disambiguate two senses of *function* and *functional structure*. Functions are special kinds of dispositions. Roughly, a component has a disposition to X in circumstances C iff it tends to X in C. Hearts are disposed to pump blood when they receive the relevant sort of electrical charges and are connected to blood vessels in the relevant sort of way and so on. Hearts are also disposed to make noise in those same circumstances. Yet not all dispositions have the same sort of explanatory significance within a larger system. A component's disposition is a function of that component iff the disposition is needed, at a certain level of abstraction, to account for the dispositions and behavior of the overall system (cf. Cummins 1983: 28–9). At the biological level of abstraction, we need to appeal to the heart's pumping blood — but not its making noise — in order to account for the dispositions and behavior of the circulatory system. Hence, all functions are dispositions, but not all dispositions are functions. To be a function is to be a disposition that plays an explanatory role in a larger system.

A functional structure of a system represents, at a certain level of abstraction, how the dispositions of each component underwrite the dispositions of the overall system. When the dispositions of the various components are manifested, they work together to explain the behavior of the overall system. The functional structure of the circulatory system would not only represent the heart's contribution, but also the contributions of blood and blood vessels, to the dispositions and operation of the circulatory system. Functional structures track how the functions of the components work together to account for the dispositions and behavior of the overall system.

Functional structures, so understood, are purely descriptive. They describe, at a certain level of abstraction, how the dispositions of the components actually work together to explain the system's dispositions and behaviors. There is no further claim that this is how the various components *should* work (together). Perhaps it is and perhaps it isn't. When a system functions improperly — when actual and proper function come apart — the purely descriptive notion tracks actual function.

There is also a normatively loaded sense of *function* and *functional structure*. To say that the function of the heart is to pump blood, in this sense, is to say that the heart is *supposed* to pump blood. This sense of *function* tracks not how things actually work but how they should work. The normatively loaded sense of *functional structure* represents, at a certain level of abstraction, the dispositions the components *should* have and how those dispositions *should* work together to underwrite the dispositions the system *should* have.

The purely descriptive and normatively loaded senses come together in properly functioning systems. If a system is functioning properly, a system's purely descriptive and normatively loaded functional structures are identical.[19]

In contrast, the purely descriptive and normatively loaded senses come apart in malfunctioning systems. If a system malfunctions, one functional structure will describe the actual organization and operation of the system and a distinct structure will describe how the system should be organized and how it should operate. Malfunction is possible only when actual (purely descriptive) functioning deviates from proper (normatively loaded) functioning.

The purely descriptive/normatively loaded distinction is *not* the narrow/wide distinction. The latter distinction concerns whether, at a certain level of abstraction, a system's environment can affect the individuation of its current structure and behavior. The former concerns whether, at a certain level of abstraction, a certain structure and behavior are (pure) descriptions of or norms for a given system. It is ordinarily assumed that norms must be individuated widely, but, in principle, purely descriptive functional structure could be wide or narrow. My view of computational structures is that the purely descriptive

19. For simplicity, I assume that computational norms require a single, specific normatively loaded structure. If proper function is compatible with a range of functional structures (as is typical in actual cases), then there will be a set of normatively loaded functional structures and the actual functional structure of properly functioning systems will be identical with one member of that set.

structure is narrow and the normatively loaded is wide. In other words, I think computational behavior is narrow and computational norms are wide. At other levels of abstraction, both behavior and norms may be wide. One externally individuated functional structure might provide the norms for our mental structure and behaviors (e.g., the norms of rationality), and — given *content* externalism — a distinct widely individuated functional structure would specify our actual mental structures and behavior.

We now have a deeper understanding of the only two differences, especially the second, between my account of computational behavior and that of Piccinini. The first difference is that my version appeals to narrow functional structure and his appeals to wide functional structure. The second difference is that my version appeals to purely descriptive structures and his to normatively loaded ones (cf., e.g., Piccinini 2015: 113–4, 151). When we get to the next section, we'll see that the second difference largely explains why Piccinini's account has a hard time identifying the computational behavior of *mal*functioning systems.

Narrowly individuated functional structure is, of course, not a good candidate to account for normatively loaded functional structure or, more specifically, the computational norms of a computing system. That is why, when I discuss the norms for a device — computational or otherwise — I follow just about everyone else in asserting that the norms are individuated widely. Miscomputation is made possible on my account, precisely because there is a gap between the narrow individuation of (purely descriptive) computational structure and the wide individuation of (normatively loaded) computational norms. A system miscomputes when its behavior manifests a narrow computational structure that the widely individuated norms say that it should not have.

My account of computational structure does impose one constraint on computational norms that I should mention. Recall that, in properly functioning systems, a system's purely descriptive functional structure is identical to its normatively loaded functional structure. In properly functioning computing systems, the system's *actual* computational structure will be identical to the computational structure that it *should* have. Since (actual) computational structures are narrowly individuated, the computational structure that a system should have must be specifiable in narrowly individuated terms.

Contrast the following two norms for a given system S:

> *Norm specified in wide terms:* When the input voltage is >5v, output the voltage that will allow the larger system to operate as an and-gate.
>
> *Norm specified in narrow terms:* When the input voltage is >5v, output >5v.

The first norm is specified in wide terms, because it references the larger system of which S is actually a component. The second norm is specified in narrow terms, because it mentions behavior that can be individuated internally to the system. There is no reference, explicitly or implicitly, to things beyond the system itself. Essentially, the internally individuated structure a system *should* have is whatever internally individuated structure properly functioning versions of that system *do* have. I say computational norms are widely individuated *not* because they are specified in terms that reference things beyond the internally individuated states and structure of the system; rather, computational norms are widely individuated because what *makes* something the proper internal structure for a system is determined by things beyond the internally individuated states and structure of the system (e.g., what it takes for the system to survive in its environment, the goal of some designer, etc.).

3. Miscomputing Externalistically: Parasitic Strategies

You have just seen how easily computational individualists can accommodate miscomputation. Since they can endorse internally individuated computational behavior while holding onto externally individuated computational norms, it is no mystery how a system can compute a function that it shouldn't. In this section, we see that Piccinini's externalism makes it more difficult to explain how such miscomputation is possible.

3.1. Teleology and Actual Computational Structure

We are focused on the sort of miscomputation in which a system computes a function when it should have computed a distinct function. Piccinini (2015b: 149, (ii)) agrees that such miscomputations exist. To adequately explain such miscomputations, our account of computational behavior must individuate the computational behavior of malfunctioning systems — not just properly functioning ones. When a system malfunctions, we need an account to determine whether the system is nonetheless computing and, if so, which function it is computing. Piccinini never explicitly provides such an account.

The heart of the problem is Piccinini's characterization of miscomputation: "if the [computing] mechanism malfunctions, a miscomputation occurs" (Piccinini 2015b: 122; cf. 14 and 2015a, sec 2.5). Fresco and Primiero make a similar mistake: "When a [computing] system fails to accomplish the purpose for which it was designed, a miscomputation can be identified" (2013: 257; cf. Coelho Mollo forthcoming, sec 4). When you think about miscomputation in this way, it is tempting to conclude that you've explained how miscomputation is possible when all you've shown is that it is possible for a computing system to issue the wrong output (2015: 148–50 and especially 24 and 148) or when all you've shown is that it's possible for a computing system to fail to follow every step of a given computational procedure (2015: 14). But to explain miscomputation, we need to explain more: we need to explain why the particular malfunctions at issue still count as computation. To miscompute is to compute.

With that said, let's see whether we can develop a workable account of miscomputation from the resources Piccinini provides. What *is* clear is that teleology should play an important role. In chapter 6, Piccinini provides an account of wide functional structure that is teleological in nature: to be a function is to make a stable contribution to attaining certain goals. Roughly, for organisms, the goal would be survival, and for artifacts, the goal would be some goal of a designer/user. This kind of functional structure is wide, because these goals are determined, in part, by something beyond the system itself.

Teleology is generally designed to play a normative role, so it's plausible that the computational structure that a system should have just is the computational structure, if any, that fulfills the system's teleological function. Externalists also sometimes insist that teleology has a significant role to play in determining (actual) computational structure.[20] Whatever this role amounts to, we must allow that the computational structure of a malfunctioning system can vary at least somewhat independently of a system's teleology.

Suppose that S1 and S2 and S3 are all computing systems of kind K, and their teleological function requires them to perform a certain computation. In particular, they are required to output 0 (any positive voltage ≤5v) when they receive input 0,0. While S1 fulfills its teleological function, S2 and S3 do not (they're broken). When given 0,0 as an input, S1 outputs 0, S2 outputs a distinct output (9v), and S3 is so broken that it doesn't output anything at all. S1 and S2 seem to be computing distinct functions, and S3 doesn't seem to be computing at all.[21] Damage can change the computational structure of a system without changing its teleology. Any appeal to teleology must respect this point.

Piccinini (2015: 109–10) claims, as is typical of those who appeal to teleology, that properly functioning systems have a kind of metaphysical priority over malfunctioning systems. He doesn't cash out this priority or its relevance for miscomputation. One way to cash it out is in terms of digit individuation: what counts as a digit for a malfunctioning system depends on what counts as a digit for properly functioning

20. See, e.g., Bontly (1998: 569–70) and Piccinini (2015: 43).

21. Why think that S3 does doesn't compute at all? Computation is a certain kind of transition between inputs and outputs: no outputs, no computation. S3 *makes as if* to compute but fails to compute. Just as throwing the ball requires the ball to leave my hand, computing requires the system to output a (complete) string of digits.

systems of its type. If a malfunctioning system takes voltages as inputs and outputs, then whether those voltages count as digits and which digits they are is determined by how, if at all, voltages are grouped in properly functioning systems of the same kind. If those voltages are digits in properly functioning systems, then they are digits in malfunctioning systems. If certain voltages count as distinct digits in properly functioning systems, then they remain distinct digits in malfunctioning systems of the same type.

In other words, we might endorse something like:

> **Parasitic Individuation:** The system's microstates (e.g., electrical charges) must be grouped together into macrostates, or digits, as demanded by proper function. If, according to these groupings, the system's current behavior involves inputs and outputs that count as digits, then the system's actual computation is given by the digits actually inputted and outputted.[22]

To see the appeal of Parasitic Individuation, consider an illustration. Let '*pf*(m) = n' represent the computation that system S should perform. Let 'af(m) = n' represent the computation actually performed by S. Suppose that, in the current circumstances, *pf*(0,1) = 1, where proper function individuates microstates into two digits, 0 (≤3v) and 1 (>5v). If S functions properly, then *af*(0,1) will likewise equal 1. Yet suppose that S is damaged in a way such that S receives the inputs 2v and 7v and outputs 2v. Parasitic Individuation tells us that the computation performed by S is *af*(0,1) = 0. Due to malfunction, we get one value for *pf*(0,1) and a distinct value for *af*(0,1). Thus, Parasitic Individuation accounts for the kind of miscomputation we are after, the kind in which a system implements one computation when it should have implemented a distinct computation. But there's baggage.

Parasitic Individuation *overcounts computations and miscomputations*. Individualists and externalists generally agree that only certain transitions between inputs and outputs count as computations. A mapping from inputs to outputs does not count as a computation if:

> a. As the system is currently composed, variations in the inputs are causally irrelevant to variations in the outputs;[23]
>
> b. As the system is currently composed, inputs cause outputs (e.g., no input charge, no output charge), but it is entirely random which output is matched with a given input.[24]

A malfunctioning computing device counts as computing, according to Parasitic Individuation, as long as the inputs and outputs are computational states *in properly functioning systems*. In the imagined case above, the damaged system receives input states (2v, 7v) and outputs states (2v), which are computational states in properly functioning systems. That was all it took to show, given Parasitic Individuation, that the imagined system was performing the computation *af*(0,1) = 0. Yet if the system is damaged so that the input states are causally irrelevant to the output states or the inputs only randomly cause the outputs, it is implausible that the damaged system is genuinely computing. For example, suppose that the mechanism that controls the outputs is stuck on 2v, so that the input charges (0,1) do *not* explain why it outputs what it does (0). The system's behavior isn't computational. While the system malfunctions, it isn't computing, and so it isn't miscomputing either. To endorse Parasitic Individuation is to overcount both computations and miscomputations in malfunctioning systems.

Yes, I remember that my individualist account is subject to overcounting problems, but Parasitic Individuation's overcounting problems are worse. Recall that there are two kinds of overcounting problems: quantitative and qualitative. A theory has the former problem when it says that a system computes some function F when it, while

22. In personal correspondence, Piccinini suggested this account to me and said that it was behind what he did say in the book.

23. See, e.g., Chalmers 2011: 326 and Piccinini 2015: ch 7.

24. See, e.g., Church 1940 and Piccinini 2015: 126–7.

eligible to compute F, is not actually computing F. A theory has the latter problem when it allows a system to compute F when the system is not even eligible to compute F, i.e., the intrinsic structure of the system is incompatible with computing F. The latter sort of problem seems to be worse. It is tantamount to a category mistake. My individualist account suffered from only quantitative overcounting — at least, there's no way for Piccinini to push a qualitative overcounting problem against me without facing one himself (§1.3). My contention is that Parasitic Individuation suffers from *qualitative* overcounting.

If a system's inputs do not cause its outputs, then there is no possible environment in which it is computing. We are working within a mechanistic theory which holds that a computational description of a system's behavior is a certain kind of explanatory description. To attribute a computation to a system is to say that the particular pattern of inputs causally explains the particular pattern of outputs. But Parasitic Individuation allows malfunctioning mechanisms to compute in the absence of such causal explanation. That's qualitative overcounting. In the next section, we'll try to fix this problem with Parasitic Individuation.

For now, I want to explain how, if I endorse a teleological account of computational norms, my individualist account can retain the popular idea that properly functioning systems have metaphysical priority over malfunctioning systems. I can do this in at least two ways. I can allow for *classificatory* and *normative* priority. Suppose we are looking at a certain device, D1, which is a properly functioning calculator of kind K. We now look at another device, D2. We might think that D2 should count as a calculator of kind K even though it fails to compute in the way demanded by the norms of K. Such a possibility raises two related questions, one normative — *Why do the norms of K apply to D2?* — and one classificatory — *Why does D2 count as a K in those circumstances?* A natural answer to these questions is that D2 bears some special relation to properly functioning members of kind K. In the case of calculators, the relation will likely involve some sort of physical similarities, similar causal histories, or even identity (imagine that D2 just is D1… after you dropped it on the floor). In the case of computing organisms, ancestry may also play a role in explaining why certain categories and norms apply to a given system even though the system isn't fulfilling the relevant norms.

Those forms of priority do not commit us to any sort of computational externalism. For the individualist can accept those forms of priority and coherently deny that there is also *individuative* priority. She can coherently deny, in other words, that the computational structure of properly functioning systems partly individuates the computational structure of malfunctioning systems. The individualist can insist that a system's *actual* computational behavior is given solely by its narrowly individuated structure while also holding that you must look outside the system to tell which norms and whether certain categories apply to it.

I treat the computational structure of all systems in exactly the same way: whether a system is functioning properly or improperly, a system's computational structure is its internally individuated functional structure (that satisfies the definitive list). That's simple. By further demanding individuative priority — by demanding that proper function partly individuates the actual computational behavior of malfunctioning systems — Parasitic Individuation individuates the computational structure of properly functioning systems one way and the computational structure of malfunctioning systems another. It pays the price of complication just to get the qualitative overcounting problem in return. Perhaps a refined version of Parasitic Individuation will fare better.

3.2. Parasitic Individuation Plus

Miscomputation is a special kind of computing malfunction. No computation, then no miscomputation either. Parasitic Individuation doesn't respect this point, as it allows malfunctioning computing systems to miscompute when they don't even have an internal structure compatible with computation. To address this problem, Parasitic Individuation must be constrained so that it types the microstates of malfunctioning systems into digits only when the malfunctioning system is eligible to compute.

Consider **Parasitic Individuation Plus**, i.e., Parasitic Individuation *plus* this constraint: a malfunctioning system computes (and so has digits) only if it has medium-independent functional structure. Parasitic Individuation Plus (PIP) is a step forward. Like Parasitic Individuation, PIP makes room for the kind of miscomputation in which a system computes one function when it should have computed a distinct function. Unlike Parasitic Individuation, it is apparently not subject to qualitative overcounting problems. Parasitic Individuation individuates the microstates of malfunctioning computing systems into digits as demanded by proper function regardless of the system's actual functional structure. PIP individuates the microstates of malfunctioning computing systems into digits (as demanded by proper function) *only when* the malfunctioning system has medium-independent functional structure. It consequently doesn't say that a system miscomputes when its inputs fail to be causally relevant to the outputs or only randomly cause the outputs. This is genuine progress.

The problem with PIP (and an additional problem with Parasitic Individuation) is that it fails to respect Medium-Independent Individuation (roughly, a system's medium-independent functional structure is what individuates its digits). For a given system, suppose that ≤3v gives us one digit (0) and >5v gives us a distinct digit (1). Given Medium-Independent Individuation, this assignment of digits tells us three things about how the system treats different voltages: it treats all voltages ≤3v the same way, it treats all voltages >5v the same way, and it treats voltages ≤3v differently than it treats voltages >5v (cf. Piccinini 127–8). Here "the same treatment" is essentially the absence of differential sensitivity. At a medium-independent level of abstraction, the system is differentially sensitive to the difference between 2v and 7v but not to the difference between 2v and 3v.

Now consider two systems, S_{PF} and S_{MF}. The two systems are of the same type and are subject to the same computational norms. These norms demand that $pf(1,1) = 0$, ≤3v count as one digit (0), and >5v as a distinct digit (1). S_{PF} is functioning properly, and its medium-independent functional structure treats voltages in the way that proper function requires. When it receives inputs 6v, 6v and outputs 3v, its behavior will be described by $af(1,1) = 0$.

In contrast, S_{MF} is malfunctioning. While it has medium-independent functional structure, it doesn't treat voltages in the way that proper function requires. It treats all voltages ≤6v the same way and all voltages >8v in a distinct way. Suppose S_{MF} receives inputs 6v, 6v and it outputs 3v. PIP individuates S_{MF}'s macrostates according to proper function — recall that the parasitic approach makes the individuation of macrostates in malfunctioning systems parasitic on the individuation of macrostates in properly functioning systems — and so PIP holds that S_{MF}'s behavior is also given by $af(1,1) = 0$. This description of S_{MF}'s behavior seems mistaken. It is true that S_{PF} and S_{MF}'s actual behavior is given by the same function, even though only the latter is miscomputing. That, by itself, is not a problem (cf. Fresco 2013, ch 2; Piccinini 2015: 13). What's problematic is that this description of S_{MF}'s computational behavior mistakenly implies that it treats 6v differently than 3v. Indeed, S_{MF} counts as malfunction precisely because it does *not* treat 6v differently than 3v. The underlying problem is that PIP (and the simpler Parasitic Individuation) doesn't respect Medium-Independent Individuation. When a malfunctioning system is genuinely computing, PIP groups microstates together, not in the way the malfunctioning system does it, but in the way that its properly functioning counterparts do it.

My individualist account fares better precisely because it respects Medium-Independent Individuation. S_{MF} treats all voltages ≤6v in the same way, and so my individualist account groups those voltages together as a single digit. When S_{MF} receives inputs 6v, 6v and outputs 3v, my account holds that S_{MF}'s behavior is given by $af(0,0) = 0$. This computational description captures the intuitive verdict that S_{MF} is miscomputing precisely because it mistypes microstates, i.e., it groups microstates into macrostates in ways that it shouldn't.[25]

4. Quasi-Individualism: The Best of Both Approaches?

My individualist account of computational behavior made explaining miscomputation a breeze. Since computational norms are individuated widely and computational behavior is individuated narrowly, it is no mystery how a system could compute one function when it should have computed a distinct function. On the downside, it avoids limited pancomputationalism only by the hair of its chinny-chin-chin. While mereological simples won't compute any function at all, solar systems will compute some function or another.

Piccinini-inspired parasitic approaches decisively avoid limited pancomputationalism. Yet Parasitic Individuation incorrectly entails that some malfunctioning systems compute when their internal structure makes it impossible for them to compute. Parasitic Individuation (Plus) incorrectly individuates which miscomputation is implemented by malfunctioning systems. Wouldn't it be great if we had a theory that got both advantages without any of the vices?

I will briefly outline such a theory for interested parties, but keep in mind that I'm happy with the individualist theory. Strict individualists claim that a system's environment (construed broadly enough to include designer intentions, any larger systems of which the system is a component, etc.) can affect *neither* whether a system computes *nor* which particular function it computes if it computes at all. My account of computational individualism is a strict individualist account in this sense. In contrast, typical externalisms allow the environment to affect *both* whether a system computes *and* which particular function the system computes. Piccinini's account is a typical externalism. Externally individuated teleological function is required for something to compute, and it helps determine which function is computed.

There is an intermediate position between strict individualism and typical externalism that we can call **quasi-individualism**. Quasi-individualism (**A**) allows the environment to affect whether a system computes at all but (**B**) doesn't allow it to affect which function the system computes, assuming the system computes at all.[26] Strictly speaking, quasi-individualism is an externalism, because it allows the environment to affect whether a system has computational structure in the first place. Yet its implications for computational individuation are probably closer to stereotypical individualist accounts than stereotypical externalist ones.

(A) enables the quasi-individualist to endorse Piccinini's explanation of why solar systems don't compute. Piccinini holds that solar systems don't compute because computation isn't one of their teleological functions. Let the quasi-individualist, then, endorse this first-pass condition: a system computes only if it has computation as a teleological function.[27] The problem with strict individualism is thus avoided.

25. One may wonder whether computational externalists should find Medium-Independent Individuation plausible after all. A certain tri-stable system is differentially sensitive to three different voltage ranges: (i) ≤2.5v, (ii) between 2.5v and 5v, and (iii) >5v (cf. Shagrir 2001). If this system is a properly functioning component of some larger system that groups (i) and (ii) together, it doesn't seem silly to treat (i) and (ii) as composing a single digit for the component system either (cf. Piccinini 2008: 229, 2015: 41). At first glance, this may seem to violate Medium-Independent Individuation, but I don't think it does. It is certainly true that the system treats (i) and (ii) differently than (iii), and the system is arguably treating (i) and (ii) in the same way at some appropriate level of generality — the level of generality at which the larger system is operating. Arguably, then, Medium-Independent Individuation is being respected. What would violate it is if we typed this tri-stable system's microstates such that ≤3v is one digit (0) and >4v is a distinct digit (1). We get the violation because the computational joints are cut where there are no medium-independent functional joints (at any level of medium-independent generality).

26. Coelho Mollo (2018) may endorse something in the neighborhood of quasi-individualism.

27. Suppose a system has narrowly individuated, medium-independent functional structure in both voltage and temperature ranges. If the quasi-individualist wants to allow that the system computes over, say, voltage ranges without computing over temperature ranges too, then she can tweak the first pass: a system computes *over a given kind of input* only if the system has a teleological function to compute *over that kind of input*.

(B) allows the quasi-individualist to individuate *which* computation is being performed in exactly the same way as the individualist. The computation is determined by the narrowly individuated functional structure (that satisfies the definitive list) — and this applies to both properly functioning and malfunctioning systems. A system miscomputes when its behavior manifests a narrow computational structure that its widely individuated computational norms say it should not have. Quasi-individualism may thus offer hope to explain miscomputation as easily and straightforwardly as the individualist while decisively avoiding limited pancomputationalism.[28]

Conclusion

This paper tries to better understand miscomputation (from within a mechanistic framework). First, it clarifies what miscomputation is and what it takes to adequately account for it (§2.1). A system miscomputes when it computes in a way that it should not compute. Thus, a complete theory of miscomputation will involve three component accounts: accounts of a system's computational behavior, its computational norms, and how a system's actual computational behavior can violate those norms.

Second, this paper develops an individualist theory of miscomputation. It defends a mechanistic account that individuates computational behavior narrowly (§1 and §2.3). All standard accounts of computational norms individuate norms widely, and any of them will do. Together these two component accounts give us the third: a system's computational behavior can violate its computational norms precisely because the former is individuated narrowly and the latter are individuated widely (§2.2).

Third, I criticized Piccinini's approach to miscomputation, which made the individuation of improper computation parasitic on the individuation of proper computation. The two versions of this approach we considered had problematic implications for what systems were computing when they were malfunctioning (§3).

Fourth, for those who are bothered by individualism's flirtation with limited pancomputationalism, I introduced quasi-individualism (§4). This view requires teleological function for a system to compute but otherwise is identical to my individualist account. Perhaps it provides the most promising approach to explaining miscomputation, but for now I'll stick with plain ol' individualism.[29]

28. Suppose that a device is eligible to compute but lacks a teleological function (and anything else that might determine what the device *should* be computing). In these circumstances, the device can compute but it can't miscompute. If you find this result odd (I don't), then you'll have an additional reason to prefer quasi-individualism over my strict individualism. For only quasi-individualism prevents computation in the absence of teleological function.

References

Allen, Collin. 2003. "Teleological Notions in Biology." *Stanford Encyclopedia of Philosophy*, Edward N. Zalta (ed.). Stable URL: http://plato.stanford.edu/entries/teleology-biology/.

Bontly, Thomas. 1998. "Individualism and the Nature of Syntactic States." *The British Journal for Philosophy of Science* 49(4): 557–74.

Chalmers, David. 2011. "A Computational Foundation for the Study of Cognition." *Journal of Cognitive Science* 12(4): 323–57.

Church, Alonzo. (1940). "On the Concept of a Random Sequence." *Bulletin of the American Mathematical Society* 46(2): 130–5.

Coelho Mollo, Dimitri. Forthcoming. "Are There Teleological Functions to Compute?" *Philosophy of Science*.

_____. 2018. "Functional Individuation, Mechanistic Implementation: The Proper Way of Seeing the Mechanistic View of Concrete Computation." *Synthese* 195(8): 3477–97.

_____. 1983. *The Nature of Psychological Explanation*. Cambridge (MA): The MIT Press.

29. Helpful comments on earlier drafts were provided by Paul Davies, Josh Gert, Matt Haug, Jonathan McKeown-Green, Marcin Miłkowski, Kevin Sharpe, multiple anonymous referees, and the audiences at Minds Online and the 2015 APA Central. Gualtiero Piccinini deserves special mention, as he provided detailed, helpful comments on multiple versions of this paper. Those earlier drafts were written thanks to a Marsden Fund Fast-Start Grant and a William & Mary Summer Research Grant. I owe these people and institutions my gratitude.

Dewhurst, Joe. 2018. "Individuation without Representation." *The British Journal for the Philosophy of Science* 69(1): 103–16.

_____. 2014. "Mechanistic Miscomputation: A Reply to Fresco and Primiero." *Philosophy & Technology* 27(3): 495–8.

Egan, Frances. 1995. "Computation and Content." *The Philosophical Review* 104(2): 181–203.

Fresco, Nir. 2015. "Objective Computation Versus Subjective Computation." *Erkenntnis* 80(5): 1031–53.

_____. 2013. *Physical Computation and Cognitive Science*. Dordrecht: Springer.

Fresco, Nir and Giuseppe Primiero. 2013. "Miscomputation." *Philosophy & Technology* 26(3): 253–72.

Piccinini, Gualtiero. 2015a. "Computation in Physical Systems." *Stanford Encyclopedia of Philosophy*, Edward N. Zalta (ed.). Stable URL: https://plato.stanford.edu/entries/computation-physicalsystems/

_____. 2015b. *Physical Computation: A Mechanistic Account*. Oxford: Oxford University Pres.

_____. 2008. "Computation without Representation." *Philosophical Studies* 137(2): 205–41.

_____. 2007. "Computing Mechanisms." *Philosophy of Science* 74(4): 501–26.

Segal, Gabriel. 1991. "Defence of a Reasonable Individualism." *Mind* 100(4): 485–94.

Shagrir, Oron. 2001. "Content, Computation, and Externalism." *Mind* 110(438): 369–400.

Sprevak, Mark. 2010. "Computation, Individuation, and the Received View on Representation." *Studies in the History and Philosophy of Science* 41(3): 260–70.