Received: 25 June 2024 | Revised: 25 September 2024 | Accepted: 30 September 2024

DOI: 10.1111/mila.12532

ORIGINAL ARTICLE

WILEY

Implementing artificial consciousness

Leonard Dung[1] | **Luke Kersten**[2]

[1]Centre for Philosophy and AI Research, University Erlangen-Nürnberg, Erlangen, Germany

[2]Department of Philosophy, University of Alberta, Edmonton, Canada

Correspondence
Luke Kersten, Department of Philosophy, University of Alberta, 2-45 Assiniboia Hall, Edmonton T6G 1C9, Canada.
Email: kersten@ualberta.ca

Funding information
LK: Killam Trusts, Grant/Award Number: RES0064987; LD: K3I-Cycling, Grant/Award Number: 033KI216

Implementationalism maintains that conventional, silicon-based artificial systems are not conscious because they fail to satisfy certain substantive constraints on computational implementation. In this article, we argue that several recently proposed substantive constraints are implausible, or at least are not well-supported, insofar as they conflate intuitions about computational implementation generally and consciousness specifically. We argue instead that the mechanistic account of computation can explain several of the intuitions driving implementationalism and non-computationalism in a manner which is consistent with artificial consciousness. Our argument provides indirect support for computationalism about consciousness and the view that conventional artificial systems can be conscious.

KEYWORDS
artificial consciousness, computational implementation, functionalism, implementationalism, substantive constraints

1 | INTRODUCTION

Driven by scientific motivations (Dehaene et al., 2017) and ethical concerns (Dung, 2023a; Metzinger, 2021; Saad & Bradley, 2022), researchers have become increasingly interested in the prospects of AI consciousness: What would it take for an AI system to be (phenomenally) conscious? One prominent strategy in this line of research is the application of computational theories of consciousness, which aim to specify the computational processes involved in consciousness (Seth & Bayne, 2022). While theories of consciousness are mainly tested against experimental

Leonard Dung and Luke Kersten contributed equally to this paper.

This is an open access article under the terms of the Creative Commons Attribution-NonCommercial License, which permits use, distribution and reproduction in any medium, provided the original work is properly cited and is not used for commercial purposes.
© 2024 The Author(s). *Mind & Language* published by John Wiley & Sons Ltd.

evidence from humans, the computational strategy seeks to identify features which are robustly, or even necessarily, tied to consciousness such that they generalize to artificial systems.

The computational strategy has proven popular given that other approaches face serious obstacles. Reasoning based on behavioral and superficial cognitive analogies to conscious humans, common in animal consciousness research (Dung, 2022; Tye, 2017), is problematic insofar as humans and AI systems are often considered too different in physical constitution and causal history to place much weight on such analogies (Shevlin, 2020). Inferences based on verbal reports, moreover, are problematic insofar as it is unclear how to elicit reports from AI systems which are genuinely introspective (Birch & Andrews, 2023).[1] Furthermore, since AI systems lack brains, using neurobiological theories of consciousness seems to be a non-starter. Thus, refining computational theories of consciousness and exploring how they can be generalized to AI systems are important research priorities.

However, despite its general promise, the computational strategy faces several challenges. First, it is not obvious that computationalism is true—that is, it is unclear that implementing the right kinds of computations is sufficient (and necessary) for consciousness (Anderson & Piccinini, 2024; Piccinini, 2020). Second, even if computationalism is true, it is an open question which computational theory of consciousness is correct, for example, global-workspace theory (Mashour et al., 2020) or perceptual reality monitoring theory (Lau, 2022). Third, even if we agree on the correct computational theory, it is not obvious what it takes to implement the requisite computations (Chalmers, 1994, 2011). In what follows, we focus on the third challenge (computational implementation), but it is worth briefly commenting on the first two.

With respect to the first challenge, as Sebo and Long (2023) note, there are two salient alternatives to computationalism. The first, known as the *biological substrate view*, says that consciousness necessarily depends on having a biological, carbon-based substrate. This view may, for example, be true if consciousness is type-identical to a certain brain state (Place, 1956; Smart, 1959). The second, known as the *biological function view* and most prominently championed by Godfrey-Smith (2016, 2020), holds that consciousness depends on fine-grained biological functions, such as metabolism, system-wide synchronization, or other functions whose implementation depends on specifics of the physical features of neurons or brain biochemistry.[2] This view is functionalist, though not computationalist (e.g., Piccinini, 2020, Chap. 14), as the functions are impossible for standard digital computers to implement.

In what follows, we say little to rule out the biological substrate or function views. However, as Sebo and Long (2023) point out, since many researchers are computationalists it is of considerable interest what computational theories entail about artificial consciousness. A recent influential report on artificial consciousness, for instance, even presupposed at the outset that computationalism is true (Butlin et al., 2023). Moreover, the preceding non-computationalist views are often conflated with the view we will discuss here: that there are substantive constraints on computational implementation. However, these are distinct views, we will suggest, supported by distinct arguments. In the following section, we will further explicate the distinction between non-computationalism and the view that there are substantive constraints on computational implementation.

With respect to the second challenge, we note that recent research is making rapid progress on the question of what different computational theories may entail for artificial consciousness (Butlin et al., 2023). The hope is that this work may inform evidence-based assessments of the

[1]Although see Dung (2023b) and Perez and Long (2023) for attempts to make questions of artificial consciousness amenable to behavioral tests.

[2]See also Cao (2022) and Seth (2021).

likelihood that different AI systems are conscious, even if we do not know which computational theory is true (de Weerd, 2024). However, without an account of computational implementation suitable for artificial consciousness questions, this research tells an incomplete story. To inform assessments of AI consciousness, we need to know both (i) what computations an AI system needs to implement to be conscious and (ii) what it takes to implement such computations. As mentioned, we will take up this second question.

To be specific, we will argue against so-called "implementationalist" views of consciousness, according to which conventional, silicon-based AI systems are not conscious because they fail to satisfy certain substantive constraints on what it takes to implement the relevant computations. Moreover, we will argue that the mechanistic account of computational implementation can account for the intuitions motivating implementationalist views, and also some non-computationalist views, in a manner consistent with computationalism and the view that conventional AI systems can be conscious. Thus, the mechanistic account indirectly supports the latter views.

The article is divided into five parts. We begin in Section 2 by surveying a recent implementationalist proposal from Shiller (2024), detailing three "integrity constraints" on artificial consciousness and the supporting arguments. In Section 3, we argue that the integrity constraints are implausible, or at least not well-supported, insofar as they are based on conflations of intuitions about computational implementation generally and consciousness specifically. We suggest that, in light of the troubles facing implementationalists in formulating constraints, researchers should look in a different direction. Thus, in Section 4, we outline three general accounts of computational implementation, arguing that the mechanistic account emerges as the most plausible view. This leads us in Section 5 to articulate two mechanistic-inspired constraints on implementation. In Section 6, we argue that these constraints capture many of the intuitions which drive non-computationalist and implementationalist views, such as Shiller's, but nonetheless refrain from supporting more substantive constraints. Consequently, our argument indirectly supports computationalism and the view that conventional AI systems can be conscious.

2 | IMPLEMENTATIONALISM ABOUT CONSCIOUSNESS

As mentioned, we understand computationalism about consciousness as the view that implementing the right kinds of computations is necessary and sufficient for consciousness. As formulated, the view expresses a rather weak claim. Since it does not entail that the nature of consciousness is exhausted by computational properties, it is even compatible with a dualist account of consciousness (Chalmers, 1996, Chap. 9). The view is favorable towards the possibility of AI consciousness because AI systems are paradigmatic examples of computing systems.

A common claim by skeptics of AI consciousness is that silicon-based systems at most "merely simulate conscious beings", rather than being conscious themselves (Wiese & Friston, 2021). As Piccinini (2021) presents the view: "[A] computational simulation doesn't actually reproduce the computations performed by the original brain—it merely represents them" (p. 139). According to this idea, computers cannot be conscious merely by virtue of running the right program; they may simulate but not implement the right computations which is why they may not qualify as conscious.

Our target here are views which, for principled metaphysical reasons, are skeptical of the possibility of consciousness in conventional, silicon-based computers. However, the views we are interested in can nevertheless endorse computationalism about consciousness, which is also

to say that they are not committed to the biological function and biological substrate views. This is a surprising combination. The underlying thought is that there are demanding constraints, not fulfilled by conventional silicon-based computing systems, which are necessary to implement, rather than merely simulate, consciousness. We will call this view, to be further unpacked later, *implementationalism* about consciousness.

To further flesh out our target, we first describe Shiller's (2024) recent view, which appears to be the most detailed and systematic example of an implementationalist account in the literature. Shiller grants that computationalism may be true, at least for the sake of the argument, but proposes three "integrity constraints" on implementation.

First, a system does not implement a computation if its degree of *material complexity* is too high. A system has a higher degree of material complexity if its functional parts and states have a more indirect relationship to the states and relations of its underlying physical components. For instance, a company has a high degree of material complexity because its states (e.g., its economic health) depend on things like ownership, balances and intellectual property, which do not have a straightforward relationship to underlying physical properties, being mediated, among other things, by complicated legal rules.

Second, a system does not implement computations if its parts are not sufficiently *causally integrated*. A system's degree of causal integration is determined by the degree to which their parts play their functional roles because of their intrinsic causal powers. For instance, the effects of bombs are produced by their internal chemistry, whereas books' effects on the world are extrinsic, that is, determined by their readers' responses. While neurons make causal contributions in virtue of their intrinsic properties, the thought-experiment below outlines a scenario in which their causal powers are extrinsic.

Third, a system does not implement computations if it is not sufficiently *continuous*. A system's continuity is determined by the degree to which its parts have persisting identities over time, which are distinct from their specific functional roles. If a system lacks continuity, its physical components may be subject to constant massive changes or replacements, so that they are only related by playing the same functional role. For example, the role of the US's commander in chief during the first world war has only been played by one person, whereas the role of vanguard has been occupied by different people at different times, making the former more continuous.

Shiller argues, convincingly to our minds, that computers built using current AI paradigms do not and cannot satisfy these three integrity constraints, although advances in neuromorphic computing might eventually allow non-conventional systems to satisfy the constraints. So, if these constraints hold, then conventional silicon-based computers cannot be conscious, in which case we do not have to worry about AI consciousness for the foreseeable future.

While Shiller expresses uncertainty about the metaphysics of consciousness, he also advances several sets of considerations in favor of the integrity constraints. Before unpacking his arguments, we need to briefly provide some background about computational implementation in general. To assess whether a system implements a computation, it seems clear we need to evaluate whether its parts are organized in accordance with the relevant computational description. However, for any system, there are many possible ways to carve it into parts (Shiller, 2024). For this reason, one possible view is that the presence of any carving which satisfies the computational description is sufficient, such that the system implements the relevant computation. However, it is not plausible to let all carvings count. As previous work has shown, such a liberal view leads to triviality (e.g., Chalmers, 2011; Putnam, 1975; Sprevak, 2018; Shiller, 2024). It entails that any system implements computations or, even more strongly, that

any system implements any computation, thus depriving computationalism of its content. So, some constraints on computational implementation are, independently of Shiller's ambitious claim, necessary merely to avoid triviality concerns—that is, not all carvings of systems which satisfy a computational description count. However, Shiller argues, if we grant some constraints on computational implementation, then this raises the question of why such requirements should be limited to the minimal number of requirements needed to rule out triviality; perhaps more constraints obtain.

Second, Shiller suggests that positing substantive constraints conforms to the spirit of computationalism and functionalism about consciousness. According to Shiller, an animating idea of functionalism is that a system is conscious if its parts interact in certain ways. In conformation to this idea, it is intuitive that the parts have to "possess some independent existence metaphysically prior to playing their roles" (Shiller, 2024). If substantive constraints ensure that this condition is fulfilled, we get a reason to believe in them.

Third, Shiller uses thought experiments to support his integrity constraints. For instance, Shiller asks us to imagine a massive brain, composed of trillions of neurons, which produce apparently arbitrary firing patterns. Additionally, there is a spreadsheet with rules for interpreting the states of each neuron and how they interact. These rules have no effect on the brain. Imagine now that "if we apply the rules included in these spreadsheets in sequence to reinterpret the state of each neuron in this brain, we would get an interpretation of the activity state of those neurons which exactly mirrors the functional dynamics of a human brain" (Shiller, 2024). Without substantive constraints on computational implementation, we may have to concede that this massive brain is conscious. And yet, arguably, it is not intuitive that such a system, whose underlying physical states only map in very indirect and intricate ways onto a computational description, is conscious. If so, substantive constraints are needed to explain why the system is not conscious.

Having outlined Shiller's view, we can now provide a general characterization of implementationalism. The view can be understood as a conjunction of the following claims:

1. Implementing some computations C is necessary for consciousness (implied by computationalism).
2. Implementing computations C is more demanding than merely simulating C since implementing computations (such as C) requires satisfying appropriate constraints.
3. Conventional AI systems fail to satisfy the relevant constraints.
4. Therefore, conventional AI systems cannot implement C, and thus cannot be conscious.

So, implementationalism holds that there are substantive constraints on computational implementation which conventional, silicon-based computers cannot fulfil. Importantly, this implies that such systems cannot be conscious regardless of the computational description they satisfy, and even if computationalism is true. At best, such systems merely simulate consciousness, in virtue of satisfying the right computational description (in the sense that there is some way, not giving rise to triviality, to carve the elements of the system so that they map onto the computational description). But the systems are not conscious, because they do not implement the computations specified in this description. Shiller's view is implementationalist, according to this formulation, because it admits the possibility of computationalism while rejecting AI consciousness in conventional computers via the integrity constraints. In Section 3, we will reject 3 and 4, while accepting 1 and 2.

We will call implementation constraints *substantive* if and only if they rule out the presence of consciousness in some cases where, if one accepts computationalism and based on some influential computational theory of consciousness, researchers would—absent these constraints—think that it is present. The typical case, of which Shiller is an example, is to rule out consciousness in conventional silicon-based computing systems, while allowing it in biological organisms or neuromorphic computing systems, which closely model biological features of human brains.

However, as we have seen, if one wants to avoid trivializing the notion of computation, some constraints on implementation are necessary. We call implementation constraints *lightweight* if and only if they are not substantive. In the typical case, they only serve to rule out triviality arguments, without affecting judgements on the distribution of AI consciousness. One can be an implementationalist about various cognitive features in virtue of postulating constraints which are substantive in that they restrict attributions of cognitive states more narrowly than what purely computational theorizing would suggest. But we focus here on consciousness.

Typically, skeptics of artificial consciousness are ambiguous regarding whether they adopt an implementationalist view, believing in substantive implementational constraints, a fine-grained functionalist view (e.g., the biological function view), thus rejecting computationalism, or both. Godfrey-Smith (2009) (not specific to consciousness), Piccinini (2021), Wiese and Friston (2021), and Wiese (2024) are relevant examples. Some of them seem to reject computationalism. Yet they can be interpreted as claiming that, even if computationalism were true, systems are not conscious if they do not satisfy a specific substantive constraint on implementation. This additional commitment would make these researchers implementationalists. Godfrey-Smith (2009) and Piccinini (2021) leave open what exactly this substantive constraint amounts to, while Wiese and Friston (2021) ground it in the free-energy principle.

Suffice it to say, we are not convinced by Shiller's arguments for implementationalism. In the next section, we will present a dilemma for Shiller's view, extend it to implementationalism in general, and then diagnose what has gone wrong.

3 | A DILEMMA FOR IMPLEMENTATIONALISM

Implementationalist views, such as Shiller's, face a dilemma, we think, owing to two possible interpretations. The first is that implementationalist accounts posit general conditions for computational implementation. That is, they state, in general terms, necessary conditions for physical systems to count as computing systems. On this interpretation, implementationalist views, such as Shiller's, apply to the same domain as accounts of computational implementation (such as those we will discuss in Section 4). On this view, we can come up with substantive constraints based on a more general connection to theories of computational implementation. This view is attractive, because it allows us to evaluate proposed substantive constraints via recourse to general, independently motivated implementation theories.

However, this first interpretation immediately runs into trouble. One reason is that all prominent theories of computational implementation (see below) are considerably more liberal than the integrity constraints permit. All prominent theories of computational implementation entail that conventional computers are implementing computations, such as common laptops or artificial neural networks (Piccinini, 2015). And yet, conventional computers, at least according to Shiller's (2024) argument, fail to satisfy the integrity constraints. Thus, on this first interpretation, the integrity constraints conflict with prominent theories of computational implementation.

Notice that this is not an accident. Rather, it is a direct product of how theories of computational implementation are developed. It is widely agreed that a plausible desideratum for accounts of implementation is that they must include paradigmatic computing systems, such as laptops, as implementing computations and exclude paradigmatic non-computing systems, such as digestive systems. Otherwise, they fail to be "extensionally adequate", that is, capture the systems we pre-theoretically want to count as computing systems (Piccinini, 2015). The trouble is that a theory, such as Shiller's (on this interpretation), which does not count conventional computers (such as laptops) as implementing computations, cannot satisfy such a condition. What this seems to suggest is that any account which is as restrictive as the integrity constraints is deeply implausible when treated as a general account of computational implementation. Moreover, notice the scope of the problem here. It applies, at least to some extent, to *all* implementationalist views, not just Shiller's. By definition, substantive constraints on computation rule out some systems as having consciousness which are (otherwise) thought to be computing systems, which brings them in conflict with the extensional adequacy desideratum.

A second interpretation is that implementationalist views like Shiller's do not propose general constraints on computational implementation but, rather, specific conditions applying only to the kinds of implementation *relevant* to consciousness. For instance, one might distinguish *proper* from *improper* implementation, wherein the former is necessary for consciousness and the latter suffices for implementing various kinds of computations not involved in consciousness.

This second interpretation, however, is implausible for two reasons. First, it is ad hoc. Since it is assumed here that computationalism may be true, nothing more than implementing the right kinds of computations may be required for consciousness. However, if no extra stuff is needed for consciousness, then it is unmotivated to say that the implementation of computations relevant to consciousness is subject to different constraints than implementation of other kinds of computations. In both cases, we are dealing with the same types of things: namely, computational states and properties.

For example, suppose one were to hold that there are different constraints on what it takes to *constitute* a house, depending on whether the house is made from wood or concrete. This would be a strange view indeed. While myriad empirical factors certainly influence when wood versus concrete parts constitute a house, it is implausible to suggest that the metaphysical relation of constitution itself works differently in both cases. There is simply no relevant *metaphysical* difference between wood and concrete. By the same token, there are no relevant metaphysical differences between consciousness and other computational properties once one grants, as implementationalists do, that computational properties may be sufficient for consciousness.

To be clear, we are not arguing that it is *impossible* to justify any substantive consciousness-specific constraints on physical processes. Rather, we are raising challenges for a specific subset of views which would need to be overcome.

Let us consider two further potential views.[3] First, one could hold that there are substantive constraints on the physical processes realizing consciousness. Lau's perceptual reality monitoring theory, for instance, entails that consciousness involves analog computation (see Lau, 2022, sections 9.5–9.7). Since standard accounts of analog computation require computational vehicles with properties that mirror their contents, it may appear that such a view would entail substantive constraints on physical realization which are specific to consciousness.

Second, one could think that there are substantive constraints on the physical processes realizing consciousness concerning having a *mind*, rather than being specific to consciousness,

[3]Thanks to an anonymous reviewer for these suggestions.

which would be independent of any demands on computational implementation. For example, possessing mental representations might require having the right kind of causal history, as teleosemantic views maintain. If consciousness requires having a mind, then this causal history constraint would apply to consciousness as well.

Consider the first view. Here it is worth distinguishing analog *modeling* and analog *computing* (Piccinini, 2015, p. 198). On the one hand, "analog modeling" refers to inferences about a target system and model using differential equations—the system of differential equations is the "analogy" between the model and the target system. On the other hand, "analog computers" refers to models of target systems which implement mathematical methods that solve systems of differential equations satisfied by a target system.

Notice that, on this interpretation of the view, something like Lau's view blends these two senses of analog computation. Strictly speaking, analog computation only concerns the functional/mechanistic properties of different classes of computers (e.g., differential analyzers). The "contents" in an analog computation do not "mirror" the contents of a target system in a semantic sense. Rather, they implement methods that solve differential equations which are analogously described in a target system. So, if one wanted to claim that analog computation, in this second sense, was necessary for consciousness, then, indeed, it would count as an implementationalist view (see condition 3). However, there would still be little reason to worry. This is because, in an important sense, digital computers can do everything general purpose analog *computers* can do. As Rubel (1989) shows, given any system of algebraic differential equations and initial conditions that describe a general-purpose analog computer, it is possible to effectively derive an algorithm that will approximate that computer's output to an arbitrary degree of accuracy. In other words, even assuming analog computing was a necessary condition for consciousness, it would still not rule out conscious digital computers.

Now consider the second view. Unlike the first view, the second view would not count as implementationalist. This is because it does not place constraints on computational implementation per se, but instead imposes implementation-independent constraints on consciousness. However, while it might be the case that there are independent reasons to think certain physical systems are or are not conscious (e.g., having the right causal history), our concern here is solely those views which grant that possession of computational states or properties may be sufficient. Thus, if the second kind of view proposes representationalist constraints on consciousness, then while we acknowledge that such constraints might hold, they are, strictly speaking, orthogonal to the present concern, which is limited to computational approaches. That being said, for what it is worth, it seems to us that most or all prominent theories of mental representation can be satisfied by some conventional AI systems (Harding, 2023; Mollo & Millière, 2023).

Moving to the second argument against consciousness-specific substantive constraints on computational implementation, such a view undermines many of the motivations initially thought to be in favor of implementationalism. Recall that Shiller argued that implementationalism conforms to the spirit of functionalism, suggesting that the parts of a system should have some independent existence prior to the functions they take part in. If the integrity constraints are not general conditions on implementation, but specific to consciousness, then it is doubtful that they can capture this intuition. This intuition seems to address a general constraint on what it takes to implement functions. It is not distinctive in any way of consciousness.[4]

[4]While Shiller (2024) is quite loose in how the notion of "intuition" is used, we think it is best understood along the lines of van Inwagen's (1997) dispositional account. Thus, intuitions here are understood as tendencies to make certain beliefs attractive; they "move" us in the direction of accepting certain propositions without taking us all the way to acceptance.

Shiller also claims that the existence of some constraints on computational implementation, required to avoid making computational implementation trivial, suggests that there may be substantive constraints on computational implementation. However, this argument fails if the integrity constraints do not address computational implementation in general. Considerations of triviality arguments do nothing to support the view that requirements on computational implementation which are specific to consciousness are needed. Indeed, given this interpretation of the integrity constraints, the integrity constraints and the requirements on computational implementation posited to address triviality concerns are about different kinds of implementation, a consciousness-specific one and a general one, and thus do not have much to do with each other.

Shiller's thought experiments invoke intuitions specific to consciousness and so are compatible with this interpretation of implementationalism. However, one may be skeptical of them on independent grounds. One reason for doubt is that different people may have different intuitions on such thought-experiments. In particular, many will be inclined to attribute consciousness to the systems Shiller describes, once they focus on the fact that such systems may be behaviorally and psychologically identical to humans, making, for example, elaborate verbal reports of their putative conscious experiences. Also, the intuitive judgements about cases Shiller relies on cannot distinguish between implementationalism and non-computationalism. Non-computationalists can explain just as well why the systems Shiller considers are not conscious, by pointing out that implementing the same computations as conscious humans is not sufficient for consciousness. In light of the theoretical problems, noted above, of motivating why one should believe in a kind of implementation specific to consciousness, the non-computationalist option seems preferable.

So, on this second interpretation, the integrity constraints are in need of a plausible motivation. It is not clear why one should posit them. We concede that AI consciousness questions are pervaded by much uncertainty and not yet scientifically tractable, so even mere plausibility considerations can have some force. But, as it stands, our impression is that there is no compelling reason at all to adopt the integrity constraints.

Moreover, these arguments extend to implementationalism in general. If substantive constraints are not interpreted as general necessary conditions on computational implementation but as specific to consciousness, then the connection between implementationalism and general views on computational implementation is severed. One cannot use general considerations about computationalism and functionalism as arguments for computationalism. Hence, on this second interpretation, it is unclear how one can present credible arguments for implementationalism. To make a robust case, implementationalists would need to show that they can do more than to appeal to intuitions about consciousness which are controversial and unstable and can also be captured by non-computationalists.

To recap, implementationalists accept that implementing the right computations may be sufficient for consciousness (computationalism) and that conventional, silicon-based systems may in principle satisfy any computational description which is a plausible candidate for a sufficient condition on consciousness. If one nevertheless wants to claim that such systems cannot be conscious, then one must say that they cannot implement the computations involved in consciousness. This may be either because they cannot implement computations at all or because they specifically cannot implement the computations relevant for consciousness, although they can implement other kinds of computations. Both routes are problematic. We do not see any viable alternative. So, the preceding dilemma seems exhaustive.

So, to escape this dilemma, one option is to abandon computationalism. This likely means accepting either a view where consciousness requires specific, fine-grained functional roles that conventional computers cannot play, such as the biological function view, or a view where consciousness is fundamentally tied to a certain kind of physical material, such as the biological substrate view. The other option is to grant that standard, silicon-based computers can be conscious, simply by virtue of implementing the right kinds of computations. Both options are incompatible with implementationalism.

The upshot, then, is that implementationalism is not viable. Our diagnosis is that the view conflates intuitions about computational implementation and non-computationalist as well as non-functionalist intuitions about consciousness. Discussions of triviality arguments and the need to restrict computational implementation somewhat are important for general accounts of computational implementation, but they do not get us to constraints which are restrictive enough to affect attributions of artificial consciousness. Intuitions about systems which satisfy computational descriptions of consciousness but, intuitively, are not conscious, are problems for computationalism, not for accounts of computational implementation. So, to the extent that one takes these intuitions seriously, they should be taken as evidence for non-computationalist views of consciousness, or perhaps even non-functionalist views, not for implementationalism. Thus, the implementationalist constraints lack a coherent rationale. They are motivated partially by non-computationalist intuitions and partially by intuitions regarding computational implementation, but both kinds of intuitions do not fit into one coherent whole.

Given the difficulties facing Shiller's requirements, and implementationalism more generally, we should only believe in lightweight constraints on implementation. In the next section, we will motivate constraints on implementation from a different direction. We outline three general accounts of computational implementation, and, in the subsequent sections, focus on one particular type of account of implementation: the mechanistic account. Then, we show that the mechanistic account can capture some of the intuitions about necessary conditions for the material substrate of consciousness which drive Shiller, as well as non-computationalist and other implementationalist views, while supporting neither implementationalism nor non-computationalism. This removes a source of support for implementationalism and non-computationalism.

4 | COMPUTATIONAL IMPLEMENTATION

The question of computational implementation (or the implementational problem) asks under what conditions it is true or false to say of a physical system that it implements a computation (Chalmers, 1994, 2011; Sprevak, 2018). It is a question of how to best explain the putative relation between physical and computational states within a physical system. Several accounts have been proposed to make sense of the relation, each with its own set of conditions.

4.1 | The causal mapping account

Historically one prominent answer is that a system must have the right causal structure to qualify as implementing a computation (Chalmers, 1994, 2011; Chrisley, 1995; Scheutz, 2001). According to this "causal mapping" account, implementing a computation requires mapping a subset of physical states of a system to a subset of computational states within a model which

support *counterfactual* relations—the counterfactual clause is supposed to rule out spurious mappings. For any computational state transition of the form $s1 \rightarrow s2$ within a system, the physical state $p1$ (which maps onto computational state $s1$) must *cause* the system to go into physical state $p2$ (which maps onto computational state $s2$). Concrete computation involves an isomorphism between physical and computational states such that it respects the causal structure of a physical process. A variant on this theme is that the mapping relation must appeal to *dispositions*. That is, for any computational state transition of the form $s1 \rightarrow s2$, if the system is in the physical state $p1$, the system will manifest a disposition to move from $p1$ to physical state $p2$, provided also that $p1$ maps to $s1$ and $p2$ maps to $s2$ (Klein, 2008); dispositional accounts are often considered "causal" insofar as dispositional properties also support counterfactual relations.

4.2 | The semantic account

A second proposal is that a physical system must manipulate representations with semantic content to count as computational (Fodor, 1981; Pylyshyn, 1984; Sprevak, 2010). According to this "semantic" view, only those physical states which qualify as representations can be mapped to computational states—this is, in addition satisfying the previous causal mapping constraints. If a physical state is not representational, then it is not computational. The motto here is "no computation without representation". An important question for the semantic account is how to specify the form a representation must take, for example, pictures, maps, and so forth. According to Fodor's (1981) view, for instance, representations must be language-like structures which support combinatorial semantics—that is, they must have the kind of syntactic structure exhibited by sentences in a language. Thus, a physical system is computational only if it involves the manipulation of language-like representations in a way that is sensitive to their syntactic structure and preserves their semantic properties. It is worth distinguishing here *implementational* views involving semantic properties and *individuation* views, where the former deals with whether semantic properties are "essential" to computation and the latter addresses whether semantic properties can be used to distinguish one computation from another (Shagrir, 2022).

4.3 | The mechanistic account

A third, more recent view is that the implementational question is best explicated within a mechanistic framework (Fresco, 2014; Kersten, 2024a; Miłkowski, 2013, 2015; Piccinini, 2007, 2015, 2020). For these "mechanistic" accounts, computational properties are mechanistic properties, computational explanation is a species of mechanistic explanation, and computational mechanisms are a special type of mechanism (e.g., ones with teleological functions). Piccinini (2015, 2020) offers a good example of the approach. For Piccinini, there are three conditions on implementation. First, a physical system must be a kind of functional mechanism—that is, a mechanism with teleological functions. The system has to possess properties that organize in such a way so as to produce or support some behavior—the reverse of which is that if a system fails to perform its function it must be the result of a breakdown in the organization of the system's component parts. Second, one of the capacities of the mechanism must be the ability to compute at least one mathematical function—a system's behavior must satisfy at least

one abstract description mapping of inputs to outputs, which also suffices to show that a system is following a rule. Third, a physical computing system must process *medium-independent vehicles*. If the input–output mapping is sensitive to at least some portion of the medium-independent vehicle over which it is defined, then it counts as a computation. According to Piccinini's version of the mechanistic account, only functionally integrated systems (mechanisms) that compute at least one abstract function via vehicle manipulation qualify as concrete computing systems.

4.4 | Assessing implementational adequacy

How should we decide between competing implementational accounts? One plausible method is to weigh up different accounts against independently plausible desiderata (Kersten, 2020; Piccinini, 2015, 2020; Sprevak, 2018). If one account does a better job accommodating a given set of desiderata than its competitors, then it offers a superior theory of implementation. Ritchie and Piccinini (2018, p. 193) offer three such desiderata:

1. *Metaphysical adequacy*: A theory of implementation should entail a fact of matter as to whether a physical system computes.
2. *Explanatory adequacy*: A theory of implementation should explain the behavior of a system in virtue of the computations it performs.
3. *Extensional adequacy*: A theory of implementation should ensure that paradigmatic cases of computing systems count as computing, and paradigmatic cases of non-computing systems do not.

Call these the "adequacy conditions".[5]

How do the causal mapping, semantic and mechanistic accounts fare with respect to the adequacy conditions? First, note that while the causal mapping account offers a form of objectivity about computation (metaphysical adequacy), in that concrete computation is a causal process of physical systems, it fails to explain the behavior of a system in terms of the procedures it executes (explanatory adequacy) or exclude non-paradigmatic cases of computing (extensional adequacy). For example, although the causal mapping account describes where a program is represented by causal transitions, it does not show how a physical system deploys programs in the production of behavior. It does not offer a computational *explanation* but a description or model (explanatory adequacy). Furthermore, many non-paradigmatic cases, such as weather or respiratory systems, will also trade in the kind of causal transitions such that they could be mapped to computational state descriptions. This means that while the causal mapping account manages to capture some instances of computing, for example, digital and analog computation, it also entails that systems that should otherwise not count as computing qualify. It is too liberal (extensional adequacy).

Second, notice that the semantic account trips up on two fronts. First, many computers execute their programs whether or not they have content, such as digital computers or automatic looms. Computing systems are sensitive to parts of computational vehicles rather than content

[5]While there are other desiderata that are sometimes invoked, such as miscomputation, these three broadly capture the majority of views on what matters for computational implementational (see, e.g., Fresco, 2014; Miłkowski, 2013; Piccinini, 2015; Sprevak, 2018).

carried by those vehicles (explanatory adequacy). Second, few would want to include all manipulations of representations as computational, such as painting a picture or recording a speech (Piccinini, 2015, p. 152). Because there is no clear way to draw the boundary around which representational manipulations count as computational, the semantic account ends up being either too liberal, including non-paradigmatic cases (e.g., painting pictures), or too restrictive, excluding extant forms of computing (e.g., analog). Either way it fails to appropriately taxonomize computing systems (extensional adequacy).

Finally, consider the mechanistic account. First, notice that only those systems which implicate functional mechanisms processing medium-independent vehicles qualify as computing systems (at least according to Piccinini's account). The mechanistic account ties implementation directly to specific features of the world. It provides a matter of fact as to whether a physical system computes (metaphysical adequacy). Second, one of the core features of the account is that a computing system must be able to produce some behavior in virtue computing a particular function. A system only implements a computation if it is the result of the activities of a mechanism's component parts. The behavior of a system is explained directly in terms of its underlying mechanistic activity (explanatory adequacy). And third, because only those functional mechanisms that process medium-independent vehicles count as computing systems, paradigmatic cases, such as Turing machines or calculators, qualify as computing systems while non-paradigmatic cases, such as digestive systems or solar systems, do not. The mechanistic account is not only sufficiently restrictive so as to be useful for explanatory purposes, but it is also liberal enough to cover a number of important types of computing, for example, Turing machines, neural networks, and digital/analog computers (extensional adequacy). The mechanistic account appears to fare better than its causal mapping and semantic counterparts.

The takeaway is that the mechanistic account offers one, if not *the*, promising theory of implementation. As we saw, it satisfies a range of demands on any account of computational implementation, offering not only a metaphysically and explanatory desirable account but one which aligns intuitively with taxonomic practices. Given this, we want to suggest that a mechanistic approach may offer a fresh starting point in thinking about the constraints facing artificial consciousness.

5 | IMPLEMENTATIONAL CONSTRAINTS

If the mechanistic account is indeed a strong candidate for an adequate theory of computational implementation, then what follows for artificial consciousness? As we have seen, the extensional adequacy conditions suggest that, contra implementationalism, no substantive constraints follow. Nevertheless, we claim that the mechanistic account motivates two lightweight constraints which may prove relevant to artificial consciousness.

5.1 | The medium-independence constraint

The first is that computations must be described in terms of the relevant *degrees of freedom* (or variation) within a physical computing system.

Recall that the mechanistic account says that concrete computations can be defined independently of the physical media that implement them (Piccinini's third condition). Computational processes and their vehicles can be said to be "medium-independent"

(Garson, 2003; Piccinini, 2015). A vehicle is medium-independent if the mapping rule that defines a computation is sensitive to different portions (spatiotemporal components) of the vehicle along specific dimensions of variation (i.e., degrees of freedom). For example, "digits" are medium-independent in virtue of the fact that a physical computing system can always distinguish different types from each other according to where they lie along the string (under normal operating conditions).

The concept of medium-independence entails that computational properties can only be implemented in physical media that possess sufficient dimensions of variation. For example, when the amplitude of an input signal to a neuron is above a certain threshold it fires. This is what is known as the all-or-one principle. Strung together over time the firing rates of a neuron produce a "neural spike train". Neural spike trains convey information across the nervous system. Importantly, though, while many aspects of neural spike trains depend on the physical properties of the medium, such as the opening or closing of ion channels, the spike timing and spike rates do not (Piccinini & Bahar, 2013). These features can be specified in terms distinct from the physical medium. They can be realized by either neural tissue or by other physical media, such as silicon-based circuits. Neurocognitive systems can be said to compute because they respect medium-independence.

By the same token, to implement computations, artificial systems need to be realized in physical media that respect medium-independence. To assess whether a physical medium can implement a computational process, the mechanistic account requires that the physical medium, such as silicon chips, preserves the right degrees of freedom. According to this first requirement, those physical systems which do not possess the requisite functional organization will fail to support computational states.

5.2 | The mechanistic constraint

The second constraint is that artificial systems must have a multi-level mechanistic structure within a physical computing system.

Recall, as previously noted, that for the mechanistic account computational explanation is a special kind of mechanistic explanation. According to Piccinini's (2015, 2020) view, for instance, it is the form mechanistic explanation takes when the activity of a mechanism can be accurately described as processing medium-independent vehicles in accordance with rules (i.e., input–output mappings). If a mechanistic level produces its behavior in virtue of the action of its computing components, it counts as a computational level. Which types of computation are performed at each level is an open empirical question.

One implication is that computational explanation requires attention to the multilevel structure of computational mechanisms. Computational explanations must not only provide abstract, functional characterization of a system, but also a detailed, structural description of how the system's component parts are organized and operate, what are respectively referred to as the "functional" and "structural" aspects of constitutive explanation (Miłkowski, 2013; Piccinini, 2015, 2020). For example, to explain horizontal eye movement, a computational explanation not only has to describe the function being computed by the ocular-motor system, such as an integration relation, but also how the neurons in the ocular-motor system carry out the particular function via preserving morphic-relations between eye-velocity and eye-position. To qualify as a case of physical computing, the ocular-motor system must be capable of sustaining a functional description in terms of an input–output relation and a structural description in

terms of the activities and organization of its component parts. A mechanism's ability to perform computations is explained mechanistically in terms of its components, their functions, and their organization.

Any description of computations will need to attend to such multi-level structure. To fully explain how a macro-state property is realized by a computational process at a lower mechanistic level, attention will need to be paid to the underlying computational components, their function, and their organization. Neither a purely information processing nor a decompositional analysis suffices for such an explanation. If a theory contains more explanatorily relevant detail about the underlying computational mechanism than some alternative, then that theory has more explanatory force, other things being equal. To be clear, the mechanistic constraint does not entail that a computational explanation must describe all the relevant details of an underlying mechanism to have explanatory force. Rather, it need only describe all the entities, activities and organizational features which prove constitutively relevant to the target phenomenon (e.g., a conscious state).

These, then, are our two implementational constraints, each derived from one or more elements of the mechanistic account. The first says that implementing computations entails preserving the requisite degrees of freedom within a system, while the second says that computational systems require a multi-level structure within the underlying computational mechanism. If computationalism is true, then these general constraints on computational implementation are also constraints on consciousness. Yet, these constraints are lightweight: If they exclude consciousness in some paradigmatic computing system, then something has gone wrong. In the next section, we show that the medium-independence and mechanistic constraints are related to Shiller's integrity constraints. While they do not imply them, they can be used to capture some of the non-computationalist and implementationalist intuitions underlying the integrity constraints. If so, this entails that a source of support for the latter views disappears.

6 | THE INTEGRITY CONSTRAINTS: REDUX

Let us return, then, to Shiller's integrity constraints.

Let us assume, for simplicity, that they, too, are intended as general constraints on computational implementation (the first interpretation from earlier). The first constraint, recall, claimed that the more indirect a system's functional properties are to the states and relations of its underlying physical components the higher degree of material complexity it possesses. A system's *material complexity* needs to be low in order to implement a computation. The second maintained that if a system's parts are not sufficiently *causally integrated*, then it cannot be said to implement a computation—a system's degree of causal integration is a function of the degree to which their parts play their functional roles in virtue of their intrinsic causal powers. The third was that a system had to be sufficiently *continuous* to implement a computation, a system's continuity is determined by the degree to which its parts have persisting identities over time. Each of the constraints was used to rule out consciousness in various cases. But, as we have seen, there are strong reasons to reject them.

Nevertheless, Shiller seems to take these constraints to capture important, independent intuitions about the necessary conditions for consciousness. Of central significance here is the general intuition that satisfying the computational description of consciousness (i.e., simulating consciousness) is not sufficient for consciousness, but that something more is required. Moreover, as the thought-experiments suggest, Shiller takes the specific constraints themselves to be intuitively attractive. These intuitions are not only relevant for implementationalists, but also

form a source of support for non-computationalism. Non-computationalists, too, take it to be intuitive that satisfying a computational description of consciousness should be distinguished from actually being conscious.

Granted, some researchers are non-computationalists for reasons unrelated to the integrity constraints—for instance, because they subscribe to a mind-brain identity theory to overcome the exclusion problem of mental causation (Kim, 2005). However, it is plausible that others believe in non-computationalism (e.g., the biological function view), and thus deny consciousness in conventional computing systems, partly because they find something like the integrity constraints intuitively attractive. If they are not implementationalists, they cannot hold that the integrity constraints describe constraints on computational implementation. However, they can nevertheless say that the integrity constraints capture something correct about the functions required for consciousness.

What we now want to suggest is that the two implementational constraints introduced in the previous section can account for the intuitive appeal of the integrity constraints, even though they do not entail the integrity constraints. The aim is to show that the various intuitions which seem to be in conflict with computationalism, and the view that conventional AI can be conscious, can, in fact, be explained in a manner consistent with these views. Thus, we present a challenge for the view that some common implementationalist and non-computationalist intuitions can be taken at face value.

First, notice that the general thrust of the point about material complexity can be captured by saying that a system's computational properties must be constrained in important ways by a physical system. For example, a transistor can be only interpreted as a logic gate when its states are assigned the values 1 and 0. This interpretation is only possible if the transistor is able to support two stable but different states. If the transistor is not bi-stable, then it conflicts with the requirements of being a computational vehicle, that is, having the requisite two degrees of freedom. It is in virtue of attending to a subset of the features of the transistor (its degrees of freedom), as they accord with computational theory, that the description of the physical transistor turns into a mathematical description of a device that processes 1 and 0 s, that is, a logic gate (Kersten, 2024a).

If one key intuition about artificial consciousness is that material complexity cannot be too high, then the notion of medium-independence naturally accommodates this fact. It contains the idea that consciousness is constrained in important ways by its underlying physical components without being a substantive constraint.

Next, consider how the idea of causal integration emerges from the mechanistic account. It is constitutive of being a computational mechanism that a mechanism's ability to perform a computation is explained directly by the activity of its underlying component parts. For example, as we saw, to qualify as a case of physical computing, the ocular-motor system must be capable of sustaining a structural description in terms of the activities and organization of its component parts. For mechanistic explanation to make sense, the component parts of the mechanism have to be causally efficacious; otherwise, behavior would not be explained by the mechanism.

Notice, though, that the mechanistic account is not committed to the specific, demanding notion of intrinsic causal powers proposed by Shiller. Instead, the relevant notion for the mechanistic account is constitution, not causality. To be specific, the relation between higher and lower-level components within a computational mechanism is one of "constitutive relevance" (Craver, 2007; Kersten, 2024b). Roughly put, the idea is that in order for a lower-level component's activity (X's ψ-ing) to be constitutively relevant to a higher-level component's activity

(S's φ-ing), one must be able to manipulate the higher-level component's activity by intervening on the lower-level component's activity (by stimulating or inhibiting), as well as the reverse. For example, to say that synaptic depolarization is constitutively relevant to action potentials, one must be able to show how changes in the distribution of neurotransmitters, such as Sodium (N^+) and Potassium (K^+), affect the occurrence of action potentials, and one must be able to show that intervening on action potentials, such as through ET, in turn affects the distribution of neurotransmitters.

For this reason, the non-integrated mechanisms Shiller envisages (e.g., a book and its readers) can implement computations on the mechanistic account, provided they are functionally isomorphic to more typical mechanisms. So, while conventional computing systems do not satisfy Shiller's constraint, they do satisfy a weaker causal constraint: They are mechanisms organized so as to perform computational functions. This means that the mechanistic account can make sense of the general observation that the physical components of an artificial conscious system have to be causally efficacious. The intuition underlying the causal integration constraint specifically is accounted for because the mechanistic account entails a similar, but lightweight, constraint.

Finally, notice that the requirements on being a computational mechanism also account for the intuitions underlying the continuity constraint. This is because to explain behavior mechanistically one must appeal to the spatiotemporal components of a mechanism and their interactions. If the components of a physical system switch or change constantly, then one cannot appeal to them in explaining behavior. If so, one would need to appeal to different components every time to account for each different instance of behavior, and none of them could be said to be involved in the "regular" production of behavior. Thus, in the simple case, mechanistic explanation requires a certain constancy in the components of the mechanism. While Shiller argues that conventional computers violate continuity, the mechanistic account classifies them as computing systems. So, if a mechanism is functionally organized in just the right way, persistence of the component parts of a mechanism is not actually required. Nevertheless, the mechanistic account explains why the continuity constraint appears attractive in the first place. It explains why different functionally organized physical systems can persist through time, carrying out different types of behaviors.

So, to summarize, it appears that the mechanistic account can accommodate the insights of each of the integrity constraints. Yet while the mechanistic account captures the intuitions which often motivate implementationalist and non-computationalist views, it does not thereby entail substantive constraints on artificial consciousness. The mechanistic account helps to explain why consciousness might be implemented in conventional artificial computing systems, such as digital or analog computers, while simultaneously accounting for various intuitions which pull in the opposite direction.

But why should there be such a close fit between the mechanistic account and Shiller's constraints? What explains the neat conceptual harmony? One important factor here, we want to suggest, is that the integrity constraints implicitly require computational systems to have multilevel structure. Computational systems, recall, are constituted by computational mechanisms with multiple levels. For a given level of a computational hierarchy C1, consisting of component parts (e.g., transistors and circuits), their function, and their organization, the components of C1 are implemented by subcomponents of level C0, such as logic gates. Moving between levels involves either abstracting from or detailing the constitutively relevant factors within the computational hierarchy. For example, abstracting from the specific dimensions and shapes of

transistors at one level might lead to a generalized description of the functional properties of a logic gate at another level.

Our suggestion is that the intuitions underlying the integrity constraints are tracking this multilevel organization. To be specific, material complexity addresses the medium-independence of computational descriptions within a mechanistic hierarchy, causal integration focuses on the relation between the activity of a mechanism's component parts and some phenomenon of interest, while continuity deals with the functional integrity of a computational mechanism over time. The intuitions underlying the integrity constraints find natural expression within the mechanistic account because it systematically captures the multilevel, functional organization of physical computing systems.

So, to sum up, we have shown that the intuitions underlying the integrity constraints can be accounted for by the lightweight constraints entailed by the mechanistic account. We have provided reasons to think that Shiller's specific integrity constraints are likely false, but that their underlying intuitions can be explained. These intuitions capture something correct, even though they do not entail the strong conclusions implementationalists envisage. Moreover, as we have seen, a natural reaction to our argument in Section 3, for opponents of consciousness in conventional computers, is to believe that the systems which Shiller describes are not conscious, but for the reason that computationalism about consciousness is false. Since our account is compatible with computationalism but accounts for the intuitiveness of Shiller's constraints, it also explains why this non-computationalist response seems attractive, even if computationalism is true. For this reason, it provides an indirect argument for computationalist views of consciousness. Computationalists have an appealing explanation of some of the intuitive resistance to their own view.

One interesting implication briefly worth mentioning is that the current proposal also opens up connections to more biologically-inspired approaches, such as the biological function and substrate views. This is because, as mentioned, computational mechanisms are special types of functional mechanisms. On Piccinini's (2015, 2020) account, for instance, computational mechanisms are mechanisms which perform at least one teleological function. For example, the neural network in the ocular-motor system responsible for horizontal eye movement computes at least one abstract function (an integration relation) in virtue of preserving the relationship between eye-velocity and eye-position (Leigh & Zee, 2006). To qualify as a physical computing system, a system must possess properties that organize in such a way so as to produce or support some function of interest—the reverse of which is that if a system fails to perform its function it must be the result of a breakdown in the organization of the system's component parts.

This is interesting because if, as Piccinini (2015, 2020) suggests, a teleological function is a stable contribution towards a goal of an organism, then goals can be either biological or non-biological. Biological goals include survival, development, reproduction or helping, whereas non-biological goals are any others pursued by an organism, such as using a Tobacco pipe to hold tobacco or crafting digital calculators to compute sums. If conscious states are tied to computational structure, then such structures may play a role in facilitating a system's biological teleological functions. Because conscious states occur spontaneously and are not subject to intentional control, the goals in question might relate to survival and development (Piccinini, 2020, pp. 319–320). There might yet be a telic role for conscious states within artificially conscious systems, assuming computational states provide a regular contribution to the goals of a system.

7 | CONCLUSION

Our aim in this article has been to articulate an account of physical computation that can productively inform investigation of artificial consciousness. We think there are three key takeaways from our analysis. First, pronouncements about artificial consciousness based on reflection about computational implementation generally should be met with skepticism or avoided if possible. Discussions of triviality arguments, for instance, are important for general accounts of computational implementation, but they are not restrictive enough to affect attributions of consciousness. Second, by accounting for intuitions driving implementationalist and non-computationalist views about consciousness based on a view consistent with computationalism, the mechanistic account provides independent support for (non-implementationalist) computationalist views of consciousness. Finally, as Piccinini (2007, 2020) intones, researchers are best served when functionalism and computationalism are pulled apart, whether when thinking about natural or artificial mentality. A good deal of the confusion surrounding artificial consciousness, we think, such as which forms of evidence or methods are relevant to the investigation, hinges on the conceptual boundaries between these two views being blurred. In pulling apart several intersecting lines of thought, we hope to have further clarified the discussion around artificial consciousness going forward.

ACKNOWLEDGEMENTS

We would like to thank Christian de Weerd, Wanja Wiese, Renee Ye, and two anonymous reviewers for helpful comments on earlier drafts of the manuscript.

DATA AVAILABILITY STATEMENT

There is no data available.

ORCID

Luke Kersten https://orcid.org/0000-0001-7054-8942

REFERENCES

Anderson, N. G., & Piccinini, G. (2024). *The physical signature of computation*. Oxford University Press.

Birch, J., & Andrews, K. (2023). What has feelings? *Aeon*. https://aeon.co/essays/to-understand-ai-sentience-first-understand-it-in-animals.

Butlin, P., Long, R., Elmoznino, E., Bengio, Y., Birch, J., Constant, A., et al. (2023). Consciousness in artificial intelligence: Insights from the science of consciousness. arXiv. https://doi.org/10.48550/arXiv.2308.08708.

Cao, R. (2022). Multiple realizability and the spirit of functionalism. *Synthese*, *200*(6), 1–31. https://doi.org/10.1007/s11229-022-03524-1

Chalmers, D. (1994). On implementing a computation. *Minds and Machines*, *4*(4), 391–402. https://doi.org/10.1007/BF00974166

Chalmers, D. (1996). *The conscious mind: In search of a fundamental theory*. Oxford University Press.

Chalmers, D. (2011). A computational foundation for the study of cognition. *Journal of Cognitive Science*, *12*(1), 323–357.

Chrisley, R. (1995). Why everything doesn't realize every computation. *Minds and Machines*, *4*, 403–430.

Craver, C. (2007). Constitutive explanatory relevance. *Journal of Philosophical Research*, *32*, 1–20. https://doi.org/10.5840/jpr_2007_4

de Weerd, C. (2024). A credence-based theory-heavy approach to non-human consciousness. *Synthese*, *203*(171), 1–26. https://doi.org/10.1007/s11229-024-04539-6

Dehaene, S., Lau, H., & Kouider, S. (2017). What is consciousness, and could machines have it? *Science*, *358*(6362), 486–492. https://doi.org/10.1126/science.aan8871

Dung, L. (2022). Assessing tests of animal consciousness. *Consciousness and Cognition*, *105*, 103410. https://doi.org/10.1016/j.concog.2022.103410

Dung, L. (2023a). How to deal with risks of AI suffering. *Inquiry: An Interdisciplinary Journal of Philosophy*, 1–29. https://doi.org/10.1080/0020174X.2023.2238287

Dung, L. (2023b). Tests of animal consciousness are tests of machine consciousness. *Erkenntnis*. https://doi.org/10.1007/s10670-023-00753-9

Fodor, J. (1981). *Representations: Philosophical essays on the foundations of cognitive science*. MIT Press.

Fresco, N. (2014). *Physical computation and cognitive science*. Springer.

Garson, J. (2003). The introduction of information into neurobiology. *Philosophy of Science*, *70*(5), 926–936. https://doi.org/10.1086/377378

Godfrey-Smith, P. (2009). Triviality arguments against functionalism. *Philosophical Studies*, *145*(2), 273–295. https://doi.org/10.1007/s11098-008-9231-3

Godfrey-Smith, P. (2016). Mind, matter, and metabolism. *Journal of Philosophy*, *113*(10), 481–506. https://doi.org/10.5840/jphil20161131034

Godfrey-Smith, P. (2020). *Metazoa: Animal minds and the birth of consciousness*. William Collins.

Harding, J. (2023). Operationalising representation in natural language processing. *The British Journal for the Philosophy of Science*. https://doi.org/10.1086/728685

Kersten, L. (2020). How to be concrete: Mechanistic computation and the abstraction problem. *Philosophical Explorations*, *23*(3), 251–266. https://doi.org/10.1080/13869795.2020.1799664

Kersten, L. (2024a). An idealised account of mechanistic computation. *Synthese*, *203*(281), 1–24. https://doi.org/10.1007/s11229-024-04526-x

Kersten, L. (2024b). Wide computationalism revisited: Distributed mechanisms, parsimony and testability. *Philosophical Explorations*, *27*(3), 280–297. https://doi.org/10.1080/13869795.2024.2332171

Kim, J. (2005). *Physicalism, or something near enough*. Princeton University Press.

Klein, C. (2008). Dispositional implementation solves the superfluous structure problem. *Synthese*, *165*, 141–153. https://doi.org/10.1007/s11229-007-9244-z

Lau, H. (2022). *In consciousness we trust: The cognitive neuroscience of subjective experience*. Oxford University Press.

Leigh, J., & Zee, D. (2006). *The neurology of eye movements*. Oxford University Press.

Mashour, G. A., Roelfsema, P., Changeux, J.-P., & Dehaene, S. (2020). Conscious processing and the global neuronal workspace hypothesis. *Neuron*, *105*(5), 776–798. https://doi.org/10.1016/j.neuron.2020.01.026

Metzinger, T. (2021). Artificial suffering: An argument for a global moratorium on synthetic phenomenology. *Journal of Artificial Intelligence and Consciousness*, *8*(1), 43–66. https://doi.org/10.1142/S270507852150003X

Miłkowski, M. (2013). *Explaining the computational mind*. MIT Press.

Miłkowski, M. (2015). Computational mechanism and models of cognition. *Philosophia Scientiae*, *18*(3), 1–14.

Mollo, D. C., & Millière, R. (2023). The vector grounding problem. arXiv. https://doi.org/10.48550/arXiv.2304.01481.

Perez, E., & Long, R. (2023). Towards evaluating AI systems for moral status using self-reports. arXiv. https://doi.org/10.48550/arXiv.2311.08576.

Piccinini, G. (2007). Computing mechanisms. *Philosophy of Science*, *4*(74), 501–526. https://doi.org/10.1086/522851

Piccinini, G. (2015). *Physical computation: A mechanistic account*. Oxford University Press. https://doi.org/10.1093/acprof:oso/9780199658855.001.0001

Piccinini, G. (2020). *Neurocognitive mechanisms: Explaining biological cognition*. Oxford University Press. https://doi.org/10.1093/oso/9780198866282.001.0001

Piccinini, G. (2021). The myth of mind uploading. In I. Hipólito, R. W. Clowes, & K. Gärtner (Eds.), *The mind-technology problem: Investigating minds, selves and 21st century artefacts* (pp. 125–144). Springer.

Piccinini, G., & Bahar, S. (2013). Neural computation and the computational theory of cognition. *Cognitive Science*, *37*(3), 453–488. https://doi.org/10.1111/cogs.2013.37.issue3

Place, U. T. (1956). Is consciousness a brain process? *British Journal of Psychology*, *47*(1), 44–50. https://doi.org/10.1111/j.2044-8295.1956.tb00560.x

Putnam, H. (1975). The mental life of some machines. In H. Putnam (Ed.), *Mind, language and reality, philosophical papers* (Vol. 2, pp. 408–428). Cambridge University Press. https://doi.org/10.1017/CBO9780511625251.022

Pylyshyn, Z. (1984). *Computation and cognition: Toward a foundation for cognitive science*. MIT Press.

Ritchie, B., & Piccinini, G. (2018). Computational implementation. In M. Sprevak & M. Colombo (Eds.), *Routledge handbook of the computational mind* (pp. 192–204). Routledge.

Rubel, L. A. (1989). Digital simulation of analog computation and church's thesis. *Journal of Symbolic Logic*, *54*(3), 1011–1017.

Saad, B., & Bradley, A. (2022). Digital suffering: Why it's a problem and how to prevent it. *Inquiry: An Interdisciplinary Journal of Philosophy*, 1–36. https://doi.org/10.1080/0020174X.2022.2144442

Scheutz, M. (2001). Causal versus computational complexity. *Minds and Machines*, *11*(4), 534–566.

Sebo, J., & Long, R. (2023). Moral consideration for AI systems by 2030. *AI and Ethics*. https://doi.org/10.1007/s43681-023-00379-1

Seth, A. K. (2021). *Being you: A new science of consciousness*. Penguin Random House.

Seth, A. K., & Bayne, T. (2022). Theories of consciousness. *Nature Reviews Neuroscience*, *23*(7), 439–452. https://doi.org/10.1038/s41583-022-00587-4

Shagrir, O. (2022). *The nature of physical computation*. Oxford University Press.

Shevlin, H. (2020). General intelligence: An ecumenical heuristic for artificial consciousness research? *Journal of Artificial Intelligence and Consciousness*, *7*, 245–256. https://doi.org/10.17863/CAM.52059

Shiller, D. (2024). Functionalism, integrity, and digital consciousness. *Synthese*, *203*(2), 47. https://doi.org/10.1007/s11229-023-04473-z

Smart, J. J. C. (1959). Sensations and brain processes. *The Philosophical Review*, *68*(2), 141–156. https://doi.org/10.2307/2182164

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

Sprevak, M. (2018). Triviality arguments about computational implementation. In M. Sprevak & M. Colombo (Eds.), *The Routledge handbook of the computational mind* (pp. 175–191). Routledge.

Tye, M. (2017). *Tense bees and shell-shocked crabs: Are animals conscious?* Oxford University Press. https://doi.org/10.1093/acprof:oso/9780190278014.001.0001

van Inwagen, P. (1997). Materialism and the psychological-continuity account of personal identity. *Philosophical Perspectives*, *11*, 305–319.

Wiese, W. (2024). Artificial consciousness: A perspective from the free energy principle. *Philosophical Studies*, *181*(8), 1947–1970. https://doi.org/10.1007/s11098-024-02182-y

Wiese, W., & Friston, K. J. (2021). The neural correlates of consciousness under the free energy principle: From computational correlates to computational explanation. *Philosophy and the Mind Sciences*, *2*. https://doi.org/10.33735/phimisci.2021.81

How to cite this article: Dung, L., & Kersten, L. (2024). Implementing artificial consciousness. *Mind & Language*, 1–21. https://doi.org/10.1111/mila.12532