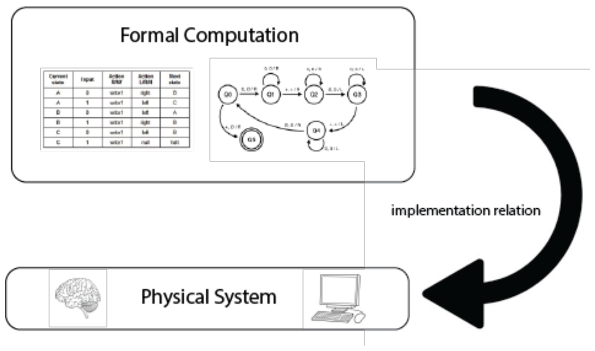
The robust mapping account and cognitive computation: a dilemma
Commentary on *The Physical Signatures of Computation: A Robust Mapping Account*

Danielle J. Williams
Washington University in St. Louis
danielle.williams@wustl.edu

01 The robust mapping account

‘Mapping views in the computation literature spell out the relation between a computational formalism and a physical system in different ways. The aim of these theories is to relate an *abstract* (as in abstracta), *computational* description to the physical world allowing us to identify which physical systems in the world literally implement computational structures. A visual representation of the relation is represented below. The mapping is meant to capture the nature of the downward arrow—the implementation relation—by specifying the features of the physical system that must be.

Any theory that does this answers the implementation question (Williams, 2023).



Some of these mappings are highly permissive in that they allow for mappings to physical systems that are obviously not computing systems (like pails of water, walls, and rocks). To fix this problem, different theories have been proposed that place various constraints on the mapping. For example, some place causal restrictions on the state transitions of the physical system. Although there has been a lot of work done to fix the mappings and prevent trivial implementations, there hasn’t been an account to goes into as much detail about the nature of the physical properties that are mapped to how to understand the computational formalism as Anderson and Piccinini’s Robust Mapping Account.

In their account, they go into detail about the states and processes that are involved in the mapping, S and P, respectively:

> **S (states):** For any [formal] computational state, there has to be at least some physical states of the system that maps onto each of the formal computational states. The physical states can't map onto more than one of the computational states.

> **P (processes):** For any [formal] computational state *transition*, the physical system must also transition to another physical state.

S and P specify an isomorphic mapping between formal states and physical states as well as formal transitions and physical transitions on a digit-by-digit basis where digits are understood as "individual computational variables" (pg. 56). S and P don't offer anything more than what traditional mapping views offer, most of which depend on some kind of isomorphic mapping of this kind. In addition to S and P, they offer a third criterion, U.

> **U (usability):** There exists a computationally transparent use plan compatible with the physical description of the physical system which specifies physical intervention protocols and conditions that would enable agents to use the physical system to compute a computation reliably for inputs of their choosing.

I won't get into the details of what makes the use plan computationally transparent. The point, I take it, is that the physical system should be able to implement every computation that is specified in the formal computation that is being mapped to the physical system without requiring additional information beyond what is implied in the formal description—that everything that's needed to perform the specified computation is present within the physical system itself. Usability constraints have been introduced into the literature by other accounts as well, they are specified more precisely here just as the other features.

What this account offers that is not present in other implementation views is the 'physical-computational-equivalence' criterion (PCE); this is an equivalence between the formal computation and the physical system where the physical states and the formal states that are being mapped should each have the same computational trajectory—this is a sameness in information. Meaning, we should be able to infer what the next physical state is for the physical system just as we are able to do with the formal computation. The PCE is meant to block trivial implementations because systems like rocks, walls, and pails of water don't have computational trajectories.

Systems that meet S and P only are considered "weak" computational systems while systems that meet S, P, U, and PCE are considered "strong." Robust computational systems, however, are only required to meet S, P, and PCE, which is the kind of computational systems that Anderson and Piccinini are interested in: hence *A Robust Mapping Account*.

According to the Robust Mapping Account, for any claim about implementation of a formal computation in physical systems to be adequate, it must be supported by a robust computational description of a physical system:

> **CDPS:** A computational description of a physical system is a physical description of a physical system that maps onto a computational definition of a computing system via a physical-to-computational mapping.

A robust physical-to-computational mapping occurs when all physical states of the physical system map onto all variables of the formal computational description and they satisfy the PCE condition. The phrase "physical-to-computational mapping" is just another way to say "implementation relation."

Note that this account stays true to the idea that the implementation relation is a relation between a formal computational description and a physical system: it answers the implementation question.

02 VR, MR, and MI

Now that the theory is on the table, I want to spend time on some things that are said in Chapter 9 (Computation and the Mind). Importantly, up to this point, the authors are giving a theory of computational implementation, which is meant to apply to computational systems generally. It is not a theory of when and how the brain, specifically, implements a computation. But because the theory should capture all computing systems, if the brain is a computer, then the theory should make sense of how the brain implements a computation. Given this, the authors address how their theory fairs when it comes to thinking about a computational mind.

In Chapter 9 the authors say that they've "expanded on their theoretical framework," however; it's not clear how what they say in this chapter is an *expansion*. The robust mapping account seems to apply to computational artifacts, as the authors show, but it's not clear, based on what is said in Chapter 9 that it can be fruitfully applied to biological computation because the "expansion" of the theory does not specify a way to understand anything like computational ***implementation*** in biological systems. (In this section I will set ‘consciousness’ aside and focus on what the authors say about cognition.)

The authors argue that cognition is explained by situated neural computations—that “there is strong evidence that there exists a computational formalism—in the sense… defined in Chapters 1-5—such that neurocognitive physical dynamics map onto a computational description couched within that formalism in ways that satisfies at least **S, P,** and **PCE**” (pg. 234).

What happens in the rest of the chapter draws heavily on previous work, especially *Neurocogntive Mechanisms* (Piccinini, 2020).

(Section 9.2, Levels, Powers, and Qualities): The authors describe how physical systems have *compositional structure*, i.e., physical systems have proper parts—or subsystems—that compose them and may in turn compose larger systems where the relation between the parts and wholes is that wholes are invariants under certain changes in their parts. For example, a forest persists through the death and birth of many of its trees, so the forest (the whole) is invariant under changes in its parts (the trees) (pg. 240). This takes higher-level properties to be *aspects* of their lower-level realizers. The difference between higher-level properties and lower-level properties is understood in terms of *granularity* (pg. 234). Higher-level properties, as I understand it, are non-structural properties such as size, mass, charge, the power to evolve, etc. (pg. 235). To return to the forest example, forest density (the higher-level property) is an *aspect* of the distribution of trees over an area (the lower-level properties) (pg. 240). *Realization*, then, according to the authors, is understood as the relation between a property of a system (forest density) and properties and relations possessed by the parts of that system (distribution of trees) such that the higher-level property (density) is an aspect if the lower-level properties (trees). They call this type of realization **variable realizability** (from Piccinini 2020, Chapter 2).

They contrast variable realizability with **multiple realizability** (MR) and say that MR occurs when a higher-level property can be variably realized—realized by many different lower-level properties—via different types of mechanisms (pg. 241). For example, mouse traps are multiply realizable because the higher-level property (catching mice) can be realized by different physical mechanisms: one-way gates, glue, electricity, etc. However, the lower-level property is constrained by whether it is the kind of thing that can realize the higher-level property.

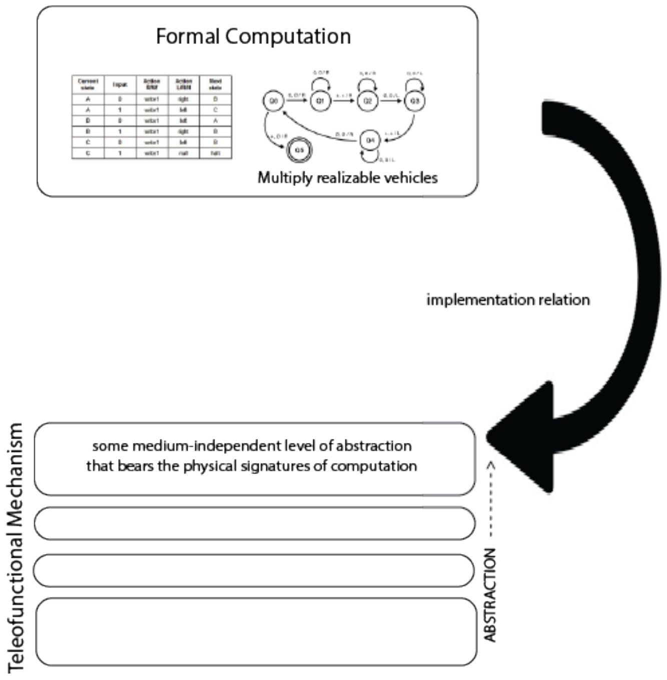
This is then contrasted with **medium-independence** (MI) where a higher-level property can be realized by different physical media but all that is required of the physical media is that it possesses enough degrees of freedom to carry out the task.

The point isn't to argue against these distinctions or how they're drawn—or whether this way of thinking about multiple realizability in philosophy of mind is correct. Instead, the point is that no matter which way you cut these characterizations, as long as the higher-level properties are understood in the way that authors understand them, they don't work with a theory of implementation. This is because the implementation relation is meant to capture the relation between a formal computation and a physical system. Higher-level properties, as they are understood in this chapter are *not* formal computations; they are aspects of the lower-level properties such as their size, shape, mass, etc. They are aspects of lower-level properties defined in terms of **abstraction** – this is different from defining the relation between *abstracta* and a physical system. Formal computations are **abstracta**. This is the first indication that the view, when it comes to cognitive computations, is going in a different direction than what we've been told up to this point.

The authors go on to say that "physical systems have different types of physical properties relevant to different phenomena defined and described at different levels of granularity and specificity. They say, "our question is which type of description captures the physical signatures of computation. The needed abstraction is the selection of a system's relevant physical states and state transitions to be mapped onto computational states and state transitions. Once the right mapping is in place, the property that manifests in genuine physical computing systems is physical-to-computational equivalence—a medium independent property" (pg. 244). I want to point out here that they still seem to be targeting the implementation relation when it comes to cognition.

Remember that the physical-to-computational equivalence (PCE) is an equivalence between the formal computation and the physical system where the physical states and the formal states that are being mapped should each have the same computational trajectory. PCE, they say, is a medium-independent property. Medium independent properties are arrived at through the process of abstraction—omitting details—so, the physical computation – the set of physical state that are equivalent to the formal computational states become evident "once the appropriate abstraction… is performed" (pg. 244). This seems to spell out the implementation relation this way where the formal computation is mapped onto some level of abstraction that bears the physical signature of computation.

The authors go on to say more—they say that **physical computation by an organism or artifact is the processing of *multiply realizable vehicles by a teleofunctional mechanism in accordance with a rule*** (pg. 248). The multiply realizable vehicles are the formal computational states, inputs or outputs which are encoded in physical forces or subsystems that are specified at some level of abstraction (the physical vehicles). Those physical forces or subsystems that encode the formal computational vehicles do so according to a rule where the rule is multiply realizable and the processing of the rule is medium-independent. This turns the implementation relation into something like this:

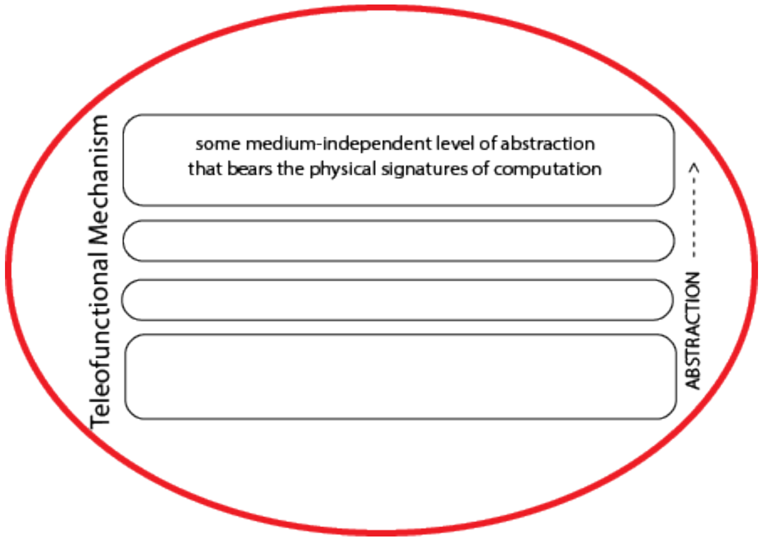

Formal Computation
Multiply realizable vehicles
implementation relation
some medium-independent level of abstraction that bears the physical signatures of computation
Teleofunctional Mechanism
ABSTRACTION

They go on to say that the issue when it comes to biological computation is determining whether a biological process maps onto a computational formalism in a robust way. This is a matter of determining whether a biological process is medium-independent considering either its *narrow* (what a system does in isolation from its environment) or *broad* (how a system relates to its environment) teleofunctions.

03 What happened to the relation?

At this stage of the argument (section 9.5.2) the authors asks whether *neurocognitive* systems perform physical computations (pg. 250). This is where the view seems to shift. Regarding narrow teleofunctions, the authors state that cognitive teleofunctions involve the production of the appropriate motor outputs in response to relevant sensory inputs and internal states while interacting with the rest of the body and the environment through the use of neurons that transmit spike trains to activate muscles. They then say that the narrow teleofunctions of neurocognitive systems are to yield outputs that stand in certain mathematical relations to inputs and internal states—that is solving mathematical problems… and aspects of spike trains that make the most functional difference are rate of transmission and timing. They conclude that spike frequency and timing are multiply realizable properties, therefore the vehicles of neural processes are multiply realizable in the right way and since the vehicles are multiply realizable, they are medium-independent, "**and biological systems whose processes are medium-independent and manipulate vehicles in accordance with a rule are biological computing systems**" (pg. 250). But remember that they are thinking of multiple realizability in terms of higher-level and lower-level properties where spike frequency and timing are higher-level aspects of the brain—they are medium independent because of this understanding. But we don't need a theory of **implementation** to relate these types of higher-level properties to the systems they belong—we come by them through *abstraction*. You can't use abstraction to get to the *mathematics*—a reason we need a theory of implementation in the first place.

What this means is that the authors have identified neurocognitive systems as *being computing systems* in virtue of their processing medium-independent vehicles in accordance with a rule. Note that this is the view put forth by Piccinini in other contexts—it's a mechanistic account of physical computation—of neurocognitive mechanisms that is meant to answer the "**individuation question**"—a question regarding how we individuate physical systems that compute and those that don't. Notice that this is not *relational*—this does not pick out computing systems by specifying the implementation relation—it's the bottom half of the implementation relation. This means that whether a system counts as a computer is decided at the mechanism-level—the individuation question is answered based on features of the physical system alone.


some medium-independent level of abstraction that bears the physical signatures of computation
Teleofunctional Mechanism
ABSTRACTION ------->

The robust mapping account is a theory of computational implementation—it answers the implementation question. This is different from the individuation question (as the authors also point out at the start of the book). However, the authors think that the individuation question and the implementation question are "two sides of the same coin" (pg. 7). Meaning, if physical computations are individuated by certain properties, then physical systems must have those properties to implement computations, and conversely if it takes certain physical properties to implement computations, then physical computing systems are individuated by those properties. Thus, the authors say, "while we will focus on implementation, our account is **also an account of computational individuation"** (pg. 7).

It follows from this that, if computing systems are individuated in virtue their processing medium-independent vehicles in accordance with a rule, then being able to process medium-independent vehicles in accordance with a rule is *enough for them to implement computations*.

04 A dilemma

However, as we know, In Chapter 5, the authors present and defend their robust mapping account, making the case that robust computational descriptions are needed to capture the physical signatures of computation—an account that must incorporate the robustness requirement (pg. 5). This means that for physical system to bear the physical signatures of computation—that for a physical system to *implement* a formal computation, it must meet criteria **S, P,** and **PCE.** And if it meets S, P, and PCE, then we know that it's a computing system, so we've answered the individuation question, too—the other side of the coin.

So, which is it? Do we get to say that a physical system implements computations when it processes medium-independent vehicles in accordance with a rule? Or is this just a special view for cognitive computation? If so, that seems strange. Or do we say that a physical system implements computations because it satisfies S, P, and PCE and *that's* what makes it a computing system, like chapter 5 says? Based on the claim made on page 1, "determining which systems compute and which systems don't is a matter of identifying the physical signatures of computation," my question is whether the physical signatures of computation differ based on whether the system is an artifact or biological, because what happens in Chapter 9 seems to provide an entirely new way to think about implementation if they really are two sides of the same coin as the authors say.