Computation is *not* essentially medium independent (draft presented at 2024 central apa)

Corey J. Maley
Purdue University
cjmaley@purdue.edu

1 Introduction

Legend has it that computation is supposed to be medium-independent (or substrate neutral, or something like this). What this means is that computation is—by definition—always characterizable without reference to any particular physical media, including the physical system that implements a given computation. The picture seems simple and uncontroversial: a computation is completely defined via an abstract automaton (e.g., a Turing Machine (TM), a finite-state automaton, a pushdown automaton, etc.), and then implemented in some physical system or other. Because they are mathematical objects, TMs and other automata have no physical properties and make no reference to physical properties. Of course a physical system that implements such an abstract mathematical object does have physical properties, but these have nothing to do with the *computation* being implemented.

The role of medium independence is illustrated clearly in Marr's celebrated "levels" of analysis for understanding information processing systems. The computational level (CL) specifies which mathematical function is being computed, and why; the representational/algorithmic level (RAL) specifies how the values of the mathematical function (and any auxiliary variables) are represented, plus the particular algorithm used to compute outputs for given inputs; and finally, the implementational level (IL) specifies how the representations and algorithms from the RAL are instantiated in a physical system. Importantly, no reference to physical media is necessary for a full specification of the CL and RAL. This is what makes it possible to create a simple calculator that computes basic arithmetic functions (CL) using binary representations of numbers and simple, digit-by-digit algorithms (RAL) using electronic circuits, marbles and wood (powered by gravity), or water moving through tubes (powered by water pressure). The CL and RAL are both medium independent, so the same system, computing the same functions, using the same type of representations and algorithms, can all be implemented in an indefinitely large variety of physical media.

This is all well and good, but I think it's wrong, for a couple reasons. First, as I've argued elsewhere, this is only true for digital computation, and not analog computation (Maley, 2021). In short, the physi-

cal properties of analog computers just are the representational properties (or, if you like, the "syntactic" properties just are the "semantic" properties). This is simply because analog computation does not abstract away from the physical properties of the system that implements that computation, whereas digital computation does exactly that. This is a specific instance of what I call the $N \neq 1$ problem of computation, where a feature of digital computation is taken to be an essential feature of *all* computation because everyone takes digital computation to be the only type of computation (or the only type worth attending to), when in fact it is a feature only of one particular type of computation (i.e., digital).

Second, insisting on medium independence as a criterion for the presence of computation makes natural computation (i.e., computation literally performed in a non-engineered system) impossible. This is simply because there is no way to infer from a natural physical system that it is implementing an abstract computation as opposed to being merely described by an abstract computation. In an artificial system, we can begin with an abstract computation and design a physical system that implements that computation. In a natural system, we do not begin with an abstract computation; we begin with the physical system, and then attempt to infer that it is computational. In order for that system to genuinely perform computations (as opposed to being describable as performing computations), we need to be able to establish from that physical system that it is implementing something medium independent. But at best, we can only establish that, at some level of abstraction, the physical system in question is multiply realizable, which is not enough for medium independence.

2 Analog representation is medium dependent

I'll assume here that analog computation is, in fact, a legitimate species of computation. Analog computing machines and devices predate digital computing machines and devices by millennia, and for several decades in the middle of the 20th century, "computing machine" would refer to an analog computer, rather than a digital one (Maley, 2023).

Perhaps the most important difference between analog and digital computation is that analog computation uses physical magnitudes to represent the magnitudes of whatever variables are being used in a computation, whereas digital computation abstracts away from physical magnitudes entirely. This is most clearly seen when we look at two similar computing systems, one digital and one analog, and analyze each using Marr's levels.

Consider two very simple computers, *D* and *A*, that only perform multiplication. They take two inputs and produce an output that is the product of those inputs. Both of these machines have the same CL characterization: they multiply their inputs, because that was these devices were designed to do.

Next, let's look at the RAL. For *D*, let's suppose that inputs and internal variables are represented as binary digital representations, and the algorithm used for multiplication is the standard (i.e., grade-school) algorithm. For *A*, let's suppose that inputs and internal variables are represented as lengths, and the mech-

anism used for multiplication is the one shown in Figure 1.

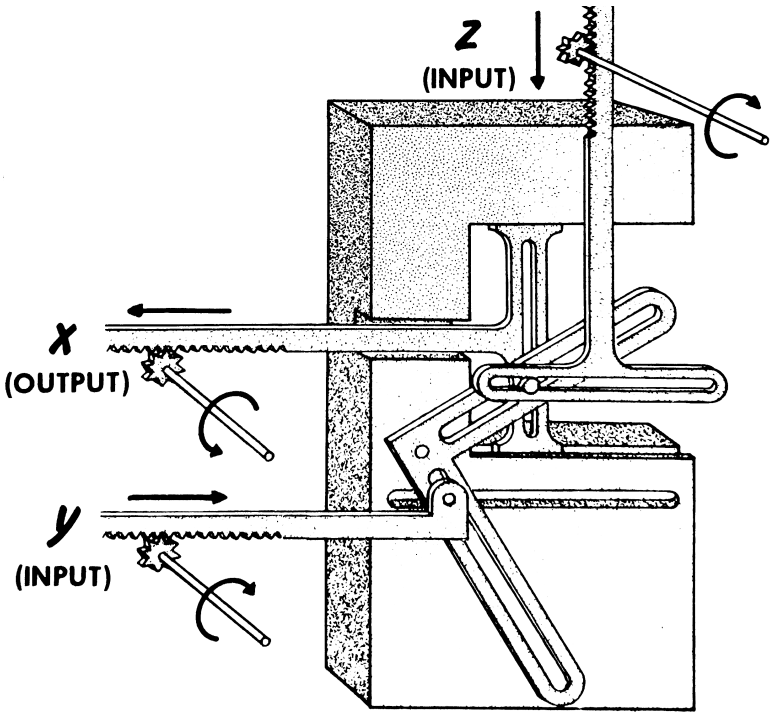

Figure 1: Mechanical multiplier.

Finally, let's look at the IL. For D, this will be a specification of how variations in some physical property map onto the two states required for the 1s and 0s of our binary representation. For example, many electronic computers use five volts to stand for a "1" and zero volts to stand for a "0". Additionally, we will need to specify the logic circuit that performs the multiplication, which will look something like the one in Figure 2 (the one shown here is only four bits, which would be a rather limited calculator, but the general idea is the same for more bits).

A different implementation for machine *D* might instead use water pressure, What about the IL for machine *A*? As argued in (Maley, 2021), the IL for machine *A* is exactly the same as the RAL. Why? Because the physical properties just are the representational properties: the lengths of the inputs and outputs in Figure 1 represent the values to be manipulated. But those very lengths are the properties relevant to the implementation of this machine. Similarly for the "algorithm," which in this case is the mechanism shown in the figure: its organization is both the way that multiplication is computed (the RAL) and the physical design of that mechanism (the IL).

As this example shows, the only place where there is no reference to physical media for machine *A* is in the CL, which is simply the mathematical specification of the function to be computed. Once we must specify *how* the inputs and outputs are represented (the RAL), we must immediately refer to a physical

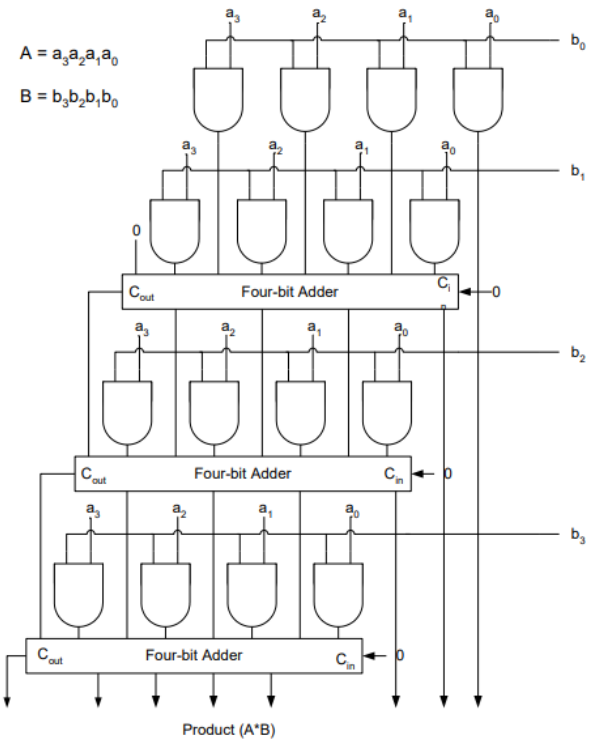

Figure 2: Four bit multiplier.

property (i.e., length). Those very same properties are the ones specified in the IL as well. So, putting this all together, we see that analog computation is medium dependent. We can characterize the mathematical function to be computed without reference to any physical media, but that is hardly enough to characterize any actual (i.e., physical) computing system.

A similar analysis is apparent in neural computation. Assume (by whatever means) we have determined that a neuron performs a simple addition of two inputs, where the inputs are always excitatory and positive, and always enough to generate a train of action potentials. In this simplified neuron, the two inputs are excitatory post-synaptic potentials (EPSPs), and the output is the frequency of the neural spike train along the axon. So, what is the CL? Addition: the output of the neuron is the sum of the inputs. What about the RAL? In other words, how are inputs and outputs represented? Inputs are represented by one type of magnitude: the magnitudes of the voltages of the EPSPs represents the magnitudes of the input. The output is represented by a different type of magnitude: the magnitude of the frequency of the spike train represents the magnitude of the output. How is this addition performed and transformed into the output? That's a bit more complicated, but would require (like the analog example above) characterizing the mechanism (i.e., the "algorithm") that combines two input voltages into another voltage that is the sum of the first two, then the mechanism that creates spikes of action potentials as a function of that voltage. And, once again, that step is also just the IL.

Now there's a place for an objection right about here, so let's address it. In the example from Figure 1, and in the neuron example, there was no mention of exactly what actual physical materials the mechanisms are made of! Is the mechanical multiplier made of wood, copper, steel, or something else? And is the neuron made of cell-goop, copper tubing, aluminum wire, or something else? Isn't *this* where the IL should come in?

I don't think so. Once we've established the physical properties doing the representing (e.g., voltages or lengths), we've place some constraints on the types of materials the relevant mechanism can be made from. Similarly for the mechanism doing the relevant operations. Wood won't work for the neuron example, because wood does not conduct electricity. Jello won't work for the multiplier in machine *A*, because it is not rigid enough. But once those constraints are accounted for, variations in the particular type of material used is irrelevant to the IL.

To see why, consider again machine *D*, the digital multiplier. Now, consider one that is implemented using copper wires and silicon, and another using gold wires and gallium arsenide. In both cases, five volts corresponds to "1" and zero volts to "0" in the binary representations used. Are these two different implementations? I don't think so: the particular materials involved aren't relevant. What would count as a different implementation would be one in which water pressure of five psi counts as a "1" and zero psi counts as a "0." Or where a lever flipped to the right counts as a "1" but flipped to the left counts as a "0." And so on.

So much for the first consideration. Let's move on to the second consideration.

3 You can't get medium independence from natural computation

Many participants in the philosophy of computation take it that there is a difference between a physical system being merely describable by (or as) some computational process, and a physical system literally performing computations. This is roughly the difference between the rather obvious fact that all sorts of physical systems and processes (virtually all of them, give or take some constraints) can be simulated computationally, but only a small handful of those systems and processes are themselves involved in actually computing (most clearly are the very systems used to perform the simulations of the other systems just mentioned); credit is due to Piccinini (2007) for first articulating this difference. In addition to the wide variety of (almost exclusively) electronic digital computational systems being created these days, many neuroscientists claim that neural systems are also computational systems. Thus, not only can neural systems be simulated computationally (as can galaxies, tornados, and virus replication), but neural systems themselves are genuine, literal, computational devices.

Now, if it's true that medium independence is required for computation, then somehow or another we should be able to establish that, whatever neural systems do, they do so in a medium-independent manner. But I don't think this can be done, because there's a dilemma underfoot. On one hand, if neural computa-

tion is analog (as I've suggested that it is, at least sometimes, if not always), then medium independence just isn't on the table for the reasons mentioned in the previous section. On the other hand, if it's not, then there is no way to "abstract" from given physical properties of neural systems to a medium independent computational characterization (e.g., a TM) that establishes that the system in question is actually implementing that characterization, as opposed to being merely described by that characterization.

I'll set aside the first concern (that neural computation is analog). This was mostly dealt with above (but eventually I should say even more). Let us skip straight to the second.

Again, remember that part of the game here is to have some criteria that distinguish physical systems that genuinely compute from those that do not (but might be describable in computational terms, which includes just about everything). Medium independence has been proposed as one of those criteria, figuring most prominently in (Piccinini, 2015, 2020), and in (Anderson & Piccinini, 2024), but also noted as a necessary feature of computation in many other accounts. For engineered computing systems, this is rather simple, as noted earlier: we start with a (by definition) medium independent characterization of a computation in the form of some mathematical automaton, then *implement* that abstract automaton in a physical system. Now, "simple" does not mean easy: real philosophical work has been devoted to understanding what it is to implement an automaton (e.g., Anderson & Piccinini, 2024; Chalmers, 1996; Rescorla, 2013; Shagrir, 2022).

The tricky part is making sense of medium independence when it comes to natural systems. If we merely want to describe a system as computing without concern for whether it really does or not, we can choose some automata and determine if it maps onto the physical system in question. So, for example, Kirkpatrick (2022) shows how we can describe both hearts and venus flytraps as computing. Why? Because there is a way to map certain macrostates of the heart and the flytrap onto particular automata. The procedure involved in doing so is similar to how one might use an automaton to model the behavior of a turnstyle, a traffic light, or most electrophysiologically active tissues. Let's look at the turnstyle, shown in Figure 3 just to keep things simple.

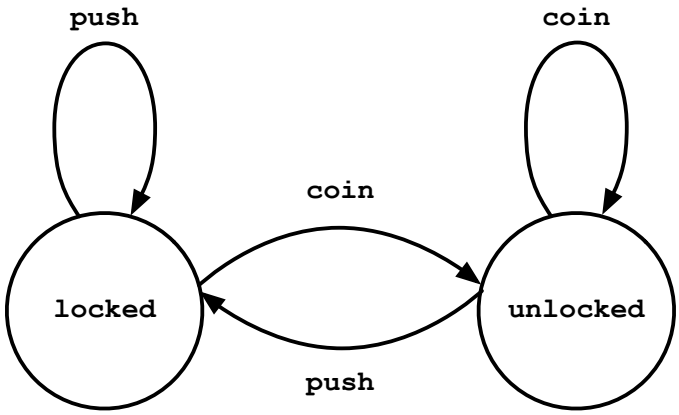

Figure 3: State diagram for a coin-operated turnstile.

There is no doubt that this state diagram nicely captures the critical behavior of a turnstile. The turnstile can be in one of two states, where different actions either result in the turnstile staying in the same state or transitioning to a different state. In a very real sense, this is an abstraction of the behavior of an actual turnstile: details about how the locking mechanism works, how a coin unlocks that mechanism, and how pushing on the unlocked turnstile results in the lock activating once again, have all been subtracted from a more detailed description of these features.

Now, is this automaton medium independent? If so, the turnstile computes, albeit a rather simple computation. That might seem a bit strange, however: a turnstile is hardly a paradigm example of a computing system. Moreover, we can perform this kind of finite state analysis with all kinds of artifacts and organisms. With enough abstraction, nearly everything will implement some automaton or another. This is not a result we want.

Perhaps this automaton is not medium independent then. After all, the labels are not arbitrary: we really mean that a coin can be inserted into the device, that it can be pushed, and that these two actions govern the transitions from it being locked or unlocked. There is no doubt that it could be multiply realized; medium independence is another matter, however.

What would be required for it to be genuinely medium independent would be yet another abstraction: relabel the states and inputs to arbitrary, uninterpreted symbols. The result (Figure 4) is standard fare in the first few chapters of any theoretical computer science textbook.

Now we have a genuine, medium independent, abstract automaton. Although very simple, we now have the kind of thing that, were it to be implemented in a physical system, would perform the computation characterized by this automaton. Fantastic! But the step from the automaton in Figure 3 to the one in 4 is illegitimate. Or, at the very least, it is certainly not a step involving *abstraction*.

This is a subtle point, but one that I think has been the source of a great amount of confusion. There is one sense of "abstract" used in these discussions which is meant to single out those mathematical ob-

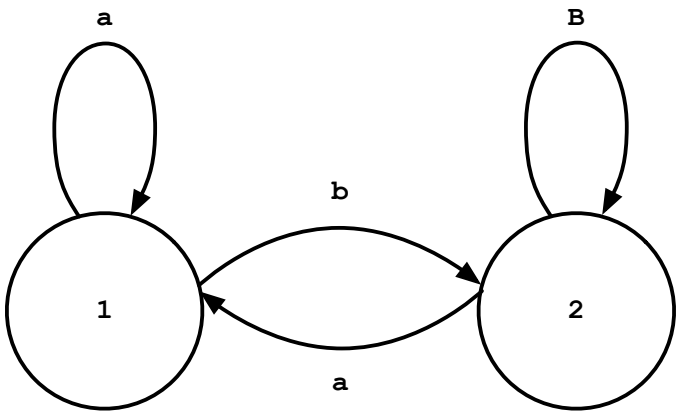

Figure 4: Abstract finite state automaton.

jects that are supposed to have some important connection to computation, such as Turing Machines, finite automata, and so on. Just like any other mathematical object, they are abstract, meaning they are non-spatiotemporal and causally inert.[1] There is a separate sense of "abstract" that figures more prominently in other discussions in the philosophy of science, in which one thing (a characterization, description, model, or something along these lines… let's call it S) is an abstraction of another (call it T) when S is obtained by subtracting away details from T.

For example, we can start with a very detailed model of neural firing, such as a Hodgkin-Huxley-based system of differential equations. These equations can account for a very large number of different potassium, sodium, and other electrical current, with contemporary versions including dozens of different ionic and leak currents. However, in some cases, we do not need all of these details, and we can subtract many of them away to get a simpler model, such as the Hindmarsh-Rose system of differential equations. This model is not nearly as detailed, but because of its simplicity, is better suited for analyzing particular types of neural behavior that is unwieldy with the Hodgkin-Huxley model. This is a case of abstraction in the second sense: details are subtracted from one model to get a simpler model, which is better suited for some particular purposes.

Note, however, that the move from calling an action a "push" to "a" is not abstraction at all: it is simply a mapping. There are no details that are subtracted from the turnstile automaton to get the one shown in Figure 4, unless we consider that literally *all* details have been subtracted. Perhaps one could think that we have subtracted everything except the fact that this device has two distinct states, and two distinct ways of transitioning between these states.

However it is that we move from the one automaton to the other (whether it is simply mapping or some incredibly austere type of abstraction), we end up with an automaton that is medium independent. Then we can implement this automaton in *all kinds* of systems. Let states 1 and 2 map to my coffee cup being

[1] If you don't like talk of abstract objects, you don't have to get off the train at this point. All that's required is that, whatever these things are, they aren't located in spacetime, and cannot figure into any causal claims.

empty or full. If it's in the full state and I drink all of the coffee from it, it's in the empty state. If it's in the full state and I fill it up, it stays full. If it's in the empty state and I drink all of the coffee from it, it stays empty. If it's in the empty state and I fill it up, it goes to the full state. Voilà! The turnstile and coffee cup perform the same computation.

This is not a result that I am happy with, and I suspect most others playing the philosophy of computation game wouldn't be happy with it either. But there is a serious tension here. When it comes to natural computation, we do not start with a pre-given automaton that we are to implement: rather, we want an account of computation that can tell us whether a system is genuinely computing or not. The fact that a system is amenable to analysis by some automata or other is irrelevant to that question: any system, abstracted enough, is amenable to a kind of finite automata, and with some mapping or questionable abstraction, can be seen as mapped to a medium independent automaton. But does such a system actually compute? Well, if medium independence is required, then this is the way to go. That strategy, however, is one that anyone can apply to nearly anything, natural or artificial, which then results in nearly everything computing.

4 Conclusion

Medium independence is not applicable to analog computation, and it cannot separate those systems that can be merely described computationally from those that legitimately perform computations. There is much more to be said here.

To mention just one, I am suspicious of the entire "implementation" relation in the first place. In the case of both a computational description of a system and a legitimately computational system, we have two things: a physical system and an abstract computational characterization. The difference between the computational system and the merely computationally-described system is supposed to be the difference between the first implementing the computation and the second being described by the computation. In both cases, this is some relation between an abstract mathematical object and a physical system. But why would one relation (i.e., implementation) to an abstract mathematical object mean that, in one case the physical object is one thing (a computer), but a different relation between the same abstract mathematical object (i.e., mathematical description) and the same physical object would make it something else (not a computation, but describable computationally)?

My own view is that all of the worrying about automata and implementation was a mistake, and that it is wrong to think of computation as primarily about automata that need to be implemented. The original sin of computation was mistaking particular mathematical models *of* computation for computation, and ignoring the fact that these models were only ever meant to model one type of computation. Instead, I think we should take computation to be primarily physical in the first place, which is what it has been, from the Antikythera mechanism to the human computers that Turing set out to model. In a slogan, I take computation to be the physical manipulation of physical representations, via processes that manipulate the

physical properties that *do* the representing. Of course, that view has issues too. But that's for another time.

References

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

Chalmers, D. J. (1996). Does a rock implement every finite-state automaton? *Synthese, 108*(3), 309–333.

Kirkpatrick, K. L. (2022). Biological computation: Hearts and flytraps. *Journal of Biological Physics, 48*(1), 55–78. https://doi.org/10.1007/s10867-021-09590-9

Maley, C. J. (2021). The physicality of representation. *Synthese, 199*, 14725–14750. https://doi.org/10.1007/s11229-021-03441-9

Maley, C. J. (2023). Analogue computation and representation. *The British Journal for the Philosophy of Science, 74*(3), 739–769. https://doi.org/10.1086/715031

Piccinini, G. (2007). Computational modelling vs. Computational explanation: Is everything a Turing Machine, and does it matter to the philosophy of mind? *Australasian Journal of Philosophy, 85*(1), 93–115. https://doi.org/10.1080/00048400601176494

Piccinini, G. (2015, January 1). *Physical Computation: A Mechanistic Account*. Oxford University Press.

Piccinini, G. (2020). *Neurocognitive Mechanisms: Explaining Biological Cognition*. Oxford University Press.

Rescorla, M. (2013). A theory of computational implementation. *Synthese, 191*(6), 1277–1307. https://doi.org/10.1007/s11229-013-0324-y

Shagrir, O. (2022). *The Nature of Physical Computation*. Oxford University Press.