*How Computation Explains*

Andrew Richmond

**Abstract.** The 20th century saw a monumental shift in our understanding of the brain, triggered by the project of *computational cognitive science*—the use of tools, concepts, and strategies from the computer sciences to investigate the brain. This is an example of *domain transfer*, like when companies apply NASA’s failure-detection strategies to ad campaigns rather than rocket components (Edsel, 2016), or when teachers “gamify” their courses, using techniques from game design to make them more engaging (Miller, 2014). Tools, strategies, or conceptual frameworks are taken from one domain and applied in another, hopefully to some advantage. The advantage, in computational cognitive science, is the provision of *computational explanations*. Philosophers have typically understood computational explanations to involve a commitment to the brain’s literally being a computer, in a sense to be specified by a metaphysics of computation. That metaphysics, by revealing what exactly we say when we say the brain is a computer, is supposed to show how and why computational explanations work, and in doing so to provide a philosophical foundation for computational cognitive science. I argue that there is no advantage to gain from this metaphysically committal approach, only confusion and awkward problems. I discuss one of the awkward problems, the triviality problem, and show that we can avoid it by rejecting the metaphysically committal approach to computational explanation. I build an alternative account of computational explanation that focuses not on the metaphysics of computation, but on the resources computational explanations bring to bear on the study of the brain. I argue that computational explanations help build perspicuous models that capture the kinds of causal structures that cognitive scientists seek to characterize, and that no metaphysics of computation is required to make sense of how they do this, or to make sense of their explanatory significance more generally.

1 Introduction

Cognitive science gives computational explanations of behavior. It tells us that the brain sees depth by computing the disparity between retinal images (Nityananda & Read, 2017), discriminates colors using cone-opponent computations (Thoreson & Dacey, 2019), localizes sounds by computing inter-aural time differences (Grothe et al., 2010), and performs reaching and grasping movements by computing vector displacements (Shadmehr & Wise, 2005). Cognitive scientists also make more general appeals to the computational capacity of neural channels or the whole brain (Gallistel & King, 2009), the computational architecture of the brain (Lake et al., 2017; Yamins & DiCarlo, 2016), and the species of computation it performs (Danks, 2019).[[1]](#footnote-1)

The question, for philosophers of cognitive science, is how and why this kind of explanation works: *what makes appeals to computation such a successful way of approaching to the brain?* What makes computational explanations *explanatory*?[[2]](#footnote-2) The received view is that computational explanations are successful and explanatory because the brain *is* a computer, and its being a computer is the reason for the capacities we’re trying to explain. E.g., it’s because the brain isa computer of a certain kind that it can support depth perception. But of course this isn’t much of an answer to our questions. If we ask why we can successfully explain depth perception by saying that the brain computes retinal disparity, it is only the beginning of an answer to say, “Because the brain is a computer that computes retinal disparity to support depth perception.” The point is to say what exactly this means—what it is to be a computer, what features the brain has in virtue of being a computer, which in turn explain its cognitive capacities.[[3]](#footnote-3) *Why* is it so important to our explanatory practices that the brain is a computer? What hinges on the brain’s being a computer? *How* does it’s being a computer underwrites our explanations? And, for that matter, how its being also want to say what it is to be any particular *type* of computer, e.g. a digital one or one computing retinal disparity. (From now on I’ll compress the two: “what it is to be a computer” means what it is to be a computer at all *and* what it is to be a particular type of computer.) It’s in this direction that answers to our original questions lie.

So upholders of the received view look for a metaphysics of physical computation: criteria that a system must meet to fall into the category computer. (Or computer of kind k, e.g., a computer that computes retinal disparity. I’ll compress these expressions: when I talk about the metaphysics of the category computer, or what it is to be a computer, I mean both the criteria for being a computer, and the criteria for being a computer of any particular type). These criteria should identify the features of the brain that make it a computer, and that therefore make computational explanations of it explanatory. Call this the *Metaphysical Approach* to computational explanation.[[4]](#footnote-4) It explicates computational explanation by giving the metaphysics of a category, computer, such that falling into that category makes a system susceptible to computational explanations.

But a satisfying metaphysics of computation is hard to come by. I’ll illustrate this by introducing the triviality problem in section 2. The main burden of this paper will be to develop an alternative approach. So in section 3 I’ll turn away from the metaphysics of computation and give an account of computational explanation that appeals only to the formal and conceptual resources computational explanations bring to cognitive scientists’ explanatory tasks. The metaphysics of computation, whatever they may be, will be irrelevant to this account: as far as computational explanation is concerned, there might as well be no such thing as a computer. Call this the *Pragmatic Approach*. I’ll develop the Pragmatic Approach in response to some objections in section 4. At that point the Pragmatic Approach will be established as a competitor to the Metaphysical Approach, and I’ll conclude in section 5 with some brief considerations (aside from the dissolution of puzzles like the triviality problem) that support the Pragmatic Approach against the Metaphysical Approach.

2 Triviality

In the examples I listed in the introduction, the brain is not merely modeled computationally, the way that the weather (e.g. Ham et al., 2019) is sometimes modeled computationally, the model being just a predictive tool—a “phenomenal model” (Craver, 2006). A computational explanation is intended to do more than predict; it is intended to explain a subject’s capacities by telling us how the subject’s brain brings them about.[[5]](#footnote-5) And according to the Metaphysical Approach, computational explanations claim that the brain brings those capacities about by *being* a computer.

The burden, then, is to say what exactly this means—what it is to be a computer, or what criteria something must meet to bea computer. Then we can see whether and how being a computer is related to the capacities that status is supposed to explain. That is, we can see how computational explanations, by appealing to the brain’s status as a computer, successfully explain its capacities. But attempts to give these criteria are plagued by a *triviality problem*. On many seemingly plausible criteria, it turns out that too many things count as computers, and computers count as computing too many things. This undermines the explanatory force of the computational approach, meaning that the success of computational explanation remains poorly understood. This section will describe the triviality problem in more detail.

It is standard to introduce triviality using the *simple mapping* account of computation (Egan, 2014), according to which a system implements a computation if the system’s states map to the stages of the computation. On this account your calculator computes addition because when you punch in any arbitrary input, e.g. “5” and then “7”, the display shows you “12”, and in doing so it has gone from internal states mapping to the numbers 5 and 7 to one mapping to the number 12.

But mappings are notoriously cheap: virtually every system maps to virtually every function or algorithm, given a suitable ‘carving up’ of the system in question. For instance, if we want a rock to compute the addition function, we need only decide which instances of addition we’d like it to perform in a span of time. If we’d like it to have just now computed 5+7=12, we take the past three seconds of the rock’s existence and map its state at each second to one of the numbers: it transitioned from state-at-second-1 (mapped to 5) and state-at-second-2 (mapped to 7) to state-at-second-3 (mapped to 12). According to the simple mapping account, this shows that the rock computes addition just as your calculator does. (Repeating states—e.g., if the rock’s next calculation includes a 5 as well—are handled by disjunctions, so it is state-at-second-1-*or*-4 that gets mapped to the number 5.) This is treated more rigorously by Putnam (1991), but the upshot is simple: mapping relations are far too numerous to constitute a definition of physical computation, because they make it too easy to be a computer. And that saps or renders mysterious the explanatory force of computation in at least three ways.[[6]](#footnote-6)

First, consider the *discovery* of computational properties in the brain. It was a non-trivial discovery that the retina performs cone-opponent computations (Jacobs, 2014). If our definition of computation entails that this is trivially true, or true given only uninformative facts about the retina, we have not understood the scientific role these computations play. If we knew ahead of time that the brain possessed all the computational properties it does, then it is mysterious how the work done to discover them was so difficult, and how it could have *confirmed* a theory of color vision.

Second, much debate in cognitive science is over *which* computations the brain performs.[[7]](#footnote-7) It is because the brain performs cone-opponent computations and not simple cone summations that it supports the kind of color vision it does (Jacobs, 2014). If it performed both computations, the connection between both models and their explananda would be severed: cone-opponent models couldn’t make a prediction about color vision that cone-summation models didn’t also make, and vice versa, because each would have to allow that the brain also performed the other model’s computations. The point is not just about prediction. It is because the brain performs certain computations and not others that those computations explain its capacities. Triviality severs the explanatory connection between computations and capacities by allowing computing systems to perform too many computations.

And third, consider systems other than the brain (or systems other than the target of a given computational explanation). Even if we find sufficiently narrow criteria so that the brain computes only a limited set of functions, it is a problem if too many other things also compute those functions. E.g., if it turns out that a rock implements an addition algorithm, then the brain’s implementing that algorithm could not be explanatory of our arithmetical abilities, because that algorithm can be implemented without supporting arithmetic. This is not to say that performing a computation must be *sufficient* for having the relevant capacity—other background features may be involved. But if those background features are computational (e.g., the computations the brain performs to use the outputs of the arithmetic module), the same problem arises: the rock has them too, on the simple mapping view. And if they are not computational, then non-computational properties do all the work making the difference between a system capable of arithmetic and a system incapable of it, and we’ve erased computation’s explanatory role. So triviality severs the explanatory connection between computations and the capacities they explain by allowing too many systems to perform any given computation.

If we’re still assuming that computational explanation is to be approached through the metaphysics of computation—criteria for a system to count as implementing computations—the problems above have to be solved by appropriately narrow criteria: ones that limit which systems implement which computations, so as to preserve the scientific role of computational explanation. This was supposed to be achieved by the causal view (Chalmers, 1996), which required not just any mapping, but one between an algorithm and a certain kind of *causal sequence* in the system in question. But this has not led to any consensus. Scheutz (2012) and Shagrir (2001) argue that the causal view still allows a problematic proliferation of computations, and Egan (2012) argues that even if it is safe from triviality, Chalmers’ definition of computation is not suitable for the use cognitive scientists put that notion to.[[8]](#footnote-8)

Other attempts have grounded computation in the teleological (Milkowski, 2013; Piccinini, 2015) or representational (Peacocke, 1994; Shagrir, 2018) properties of computational systems. But teleological properties are hotly debated, and incur severe explanatory debts themselves, so it’s not clear that they are appropriate for this purpose (cf. Dewhurst, 2018).[[9]](#footnote-9) There are troubling problems for most views of representation too (e.g., see Egan, 2019), even if they are less worrying than the problems for teleological properties. Of course, just noting these problems does not constitute a serious argument against these other views. Failed attempts and difficult problems indicate challenges, not reasons for surrender. The hero is not defeated by the mere sight of skeletons strewn outside the dragon’s lair, nor by the suggestion of stubborn obstacles ahead. Nonetheless, the hero would surely appreciate our pointing out an alternative approach to her journey—one with fewer dragons and less stubborn obstacles.

I have just such an approach to point out. To begin with, I will focus on computational explanation itself, rather than jumping straight to the metaphysics of computation. Egan (2012) and Shagrir (2006) begin in roughly the same way, and I argue explicitly for such an approach in [redacted for blind review]. But Egan and Shagrir still go on from this starting point to develop a metaphysics of computation—one informed by a deeper look at the practice of computational explanation itself, but no less susceptible to the problems I’ve discussed. I will argue that we do not need a metaphysics of computation at all: a careful look at how computational explanation works reveals that it does not require its target systems to meet membership for the category computer—it doesn’t even require that such a category exists. A view that eschews the metaphysics of computation, not just as a starting point but on the whole, takes a *Pragmatic Approach* to computational explanation.

3 The Pragmatic Approach

A simple way to dissolve the triviality problem is to deny that an account of computational explanation requires criteria for the category computer. If we need no such criteria, there can be no problem of our criteria being too encompassing. The burden of this section is to develop an approach to computational explanation that has no need for a category of computer, or criteria for membership in it.

First, let me draw a distinction that the last section failed to make. I distinguished between mere predictive models and genuine computational explanations. One way of making this distinction more precise is to say that genuine computational explanations give *process models* of their target systems—models ofthe processes that generate their behavior. Simon and Newell (1973) expressed this early on:

We do not say that we understand the magic [trick] because we can predict that a rabbit will emerge from the hat when the magician reaches into it. We want to know how it was done—how the rabbit got there. Programs like LT [the authors’ “Logic Theorist”] are explanations of human problem-solving behavior only to the extent that *the processes they use* to discover solutions are *the same as the human processes*. (p. 147, my italics)[[10]](#footnote-10)

So there is the distinction between *merely predictive models*, like computational models of the weather tend to be, and *process models*, which detail the processes a system undergoes to generate its outputs. But process models are not necessarily models that attribute, to their target systems, membership in some category. Consider a model of a two-body system expressed in calculus equations, or the famous models of fluid movement that can be applied to traffic jams. Such models can be more than predictive—they can model the processes that bodies or traffic go through to generate their dynamics or final positions. So these can be process models. But though a model is couched in calculus, the two-body system it models need not be a ‘*calculizer*’. Nor, for that matter, does traffic have to be a fluid.

So the missing distinction is this: for a category—like calculizer, fluid, or computer—it is possible that a process model explains a target system *because it subsumes the target system under that category*. In that case, to understand the model’s explanatory role we must understand this category and its membership conditions. But it is also possible that a process model explains a target system without subsuming the system under the category; it is possible that the category needn’t even *exist*. Then the membership criteria for that category have no role to play in our understanding for the model’s explanatory role. Such is the case for the category calculizer in calculus models of two-body systems, and such is the case for the category fluid in models of traffic borrowed from fluid dynamics, or for the category virus is models of disinformation as a sort of virus. These types of models all work because they bring powerful modeling and conceptual tools to the understanding of their subject matter. And they are appropriately applied to their target systems not because those target systems satisfy a metaphysics calculizing, virusing, or fluiding, but because the target systems have certain features that make the models particularly useful with respect to the modelers’ explanatory goals.

The Metaphysical Approach takes computational cognitive science to trade in process models whose success must be understood, and underwritten, by an understanding of the criteria for membership in the category computer. The Pragmatic Approach takes computational cognitive science to trade in process models whose success can be perfectly well understood, and underwritten, without talking at all about criteria for the category computer. With two caveats, this is the space the view below will work in. The first caveat is that there are deflationary, or “metaphysically light” (Egan, in conversation) understandings of the category computer on which it is hard to deny that there is such a category, or that systems that receive computational explanations fall into it. I’ll return to this in Section 4. The second caveat is that even if computational explanations are just process models, they are not *just* process models—they have distinctive and interesting features that it is important to recover. And those features will be recovered, on the account to come.

OK, enough stage-setting. The key to understanding the success of computational explanation on the Pragmatic Approach will be to see how it serves the broader goals of cognitive science. Cognitive science’s target explananda are the cognitive capacities of complex systems like biological organisms: the capacity to detect and distinguish between stimuli, to decide on a course of action, to navigate spatial and social environments, and so on.[[11]](#footnote-11) These are capacities to produce appropriate outputs from a range of inputs. And cognitive science, since the rejection of behaviorism, aims to explain these capacities, or these input–output structures, by appeal to the internal causal structure of the system in question.[[12]](#footnote-12) It is this structure that mediates the system’s behavior; it is this structure that determines its solutions to the problems it faces; and it is this structure in terms of which we understand that behavior and those solutions.[[13]](#footnote-13)

These goals call for a description, at an appropriate level of grain, of the brain’s causal organization and processes.[[14]](#footnote-14) This means we need, at least:

**(A)**  Some language or formalism in which to describe the causal structures that underly cognitive capacities.

**(B)**  Conceptual resources with which to form and understand hypotheses, couched in that language, about those causal structures and their relationship to the capacities they support.

**(C)**  Heuristics and background knowledge that make forming, understanding, and testing hypotheses easier.

The need for good languages or formalisms, as in (A), is widely discussed. See, e.g., Lazebnik (2004) on the importance of a good language in biology for making predictions, framing hypotheses, revealing important features of the target system and making them salient, and providing unity to a field. But not just any language, even a highly descriptive one, will do, and (B) is required to ensure that language is *functionally appropriate* to our subject—it should facilitate useful descriptions of the particular causal structures we seek (Lazebnik, 2004). (C) is more nebulous, but an important part of any research program—the descriptions, predictions, and explanations just described have to be cognitively tractable and facilitate further investigation. What (A)–(C) would give us is a formalism in which we can describe the internal causal structures that cognitive science seeks (from A), see how those structures bring about the capacities under investigation (from B), and efficiently and fruitfully theorize about those capacities (from C).[[15]](#footnote-15) It is also crucial that our formalism and conceptual resources make it possible to *test* our theories, but this is essentially the problem of giving them empirical content—I’ll treat that in the next section.

My claim is that computation provides a set of formalisms and conceptual resources satisfying (A)–(C), and as such it contributes to the goals of cognitive science—it provides resources to explain how the internal structure of a system brings about its cognitive capacities. If it does this well, it will facilitate good explanations. And if these explanations hinge on good descriptions of causal structures—not on the subsumption of a target system under the category computer—then there need be no criteria for membership in that category, any more than models using calculus require there to be criteria for membership in the category calculizer. This would vindicate the Pragmatic Approach to computational explanation. The task, then, is to see how (A)–(C) are satisfied.

Let’s begin with (A). We need a formalism in which to capture the kinds of causal structures cognitive scientists seek—causal structures that, in the brain, explain cognitive capacities. There is likely no unique formalism appropriate to this task, and, for that matter, it is unclear how formalisms should be individuated. In particular, what counts as a *computational* formalism is not straightforward, and seems to depend on how tools from computer science are exported into new domains (Smith, 1999). The formalism of Turing machines is used in computational explanations, as are the formalisms of finite state automata and combinatorial state automata, along with the formalisms involved in describing perceptrons and artificial neural networks, Python or MATLAB programs, wiring diagrams (e.g. Sejnowski et al., 1988), arithmetical operations (e.g. Devalois & Devalois, 1993), calculus equations (e.g. Shadmehr & Wise, 2005), statistical functions (e.g., when a neuron is described as computing a Laplacian of Gaussian function, Egan, 1999, p. 192), etc.[[16]](#footnote-16) So I won’t rigorously define “computational formalism”; instead I will lean on the way the notion of computation is used in cognitive science, and allow that whatever, for cognitive scientists, counts as a computational description, *is* a computational description. The task is to ask what those descriptions have in common that makes them suited to achieving (A)–(C).

What the formalisms above have mostly in common is that they invoke the devices built by computer engineers (e.g., wiring diagrams), the programs designed by computer programmers (e.g., MATLAB programs or neural networks), or the mathematical structures investigated in computer science (e.g., Turing machines or finite state automata). In using these formalisms, cognitive science describes the brain in terms borrowed from the science, engineering, or programming of computers. I’ll condense this by saying it uses formalisms borrowed from *computing disciplines*.[[17]](#footnote-17) I’ll call these formalisms *computational formalisms*, and they are the formalisms that I propose computational explanation brings to bear on (A). The conceptual resources that allow computational explanation to meet (B) and (C) are conceptual resources drawn from computing disciplines, or tied closely to computational formalisms.[[18]](#footnote-18)

The case to be made is that computational formalisms and computational conceptual resources serve (A)–(C) well. One important feature of computational formalisms is their facility with *functional abstraction*. Functional abstraction highlights an aspect of a system component, usually described mathematically, that captures its contribution to the system’s behavior at a higher level of abstraction. A paradigmatic example is naming high electron flow in a wire “1”, and low electron flow “0” (Hillis, 1998, pp. 18–19). Any variation in the electron flow within either “1”-signals or “0”-signals disappears, along with the gradient between “1”- and “0”-signals, and all the wire’s other features. In fact, *the wire itself* disappears. All that remains is the distinction we’ve selected as significant for our purposes—a distinction between 1 and 0. Because we’ve chosen a description of the wire under which it behaves predictably (we know the circumstances that will put the wire in a 1-state and the circumstances that will put it in a 2-state), we can exploit that distinction to build more complex functions like logic gates.[[19]](#footnote-19) There is no need to belabor the utility of functional abstraction for engineering, but it is important that it offers benefits in the *reverse*-engineering of the brain as well, particularly in a computational context. The saltatory action potential, e.g., lends itself well to a characterization in terms of 1s and 0s; this was an explicit motivation for von Neumann’s (1958) and McCulloch and Pitts’ (1943) treatment of the brain in computational terms.

The story is now, of course, much more complicated. We don’t treat neurons as logic gates but as (something at least as complex as) non-linear functions of weighted sums of inputs. As a result, the functions neurons compose in our models are more complex than truth functions. But it is still a major goal of cognitive science to “decompose cognition into functional components”, and to discover how the brain’s activity at an “elementary” or neural level can be characterized so as to compose those functions (Kriegeskorte & Douglas, 2018). And to do this we need a way of abstracting from the complex causal profile of neurons (or ensembles of them, or brain areas) to well-understood mathematical functions. To be clear, functional abstraction is an unavoidable feature of a mathematical description of any physical system. The question is which mathematical formalisms to use. And what better formalisms than the ones for which we have the relevant implications of their descriptions are well-understood? We know a great deal about the processes defined by computational formalisms: how fast a process is, how many steps it takes, how it scales to different inputs, how efficient it can be at what cost to accuracy, what it can do if it includes recurrent steps and if it doesn’t, and so on (see, e.g., Kriegeskorte & Douglas, 2018, Box 3), and these are many of the same questions we have about the brain.

Computational formalisms are also useful because they enable us to give perspicuous descriptions in terms of algorithms and hierarchies. *Algorithms* are functions strung together into sequences. These provide a clear and intuitive way of connecting a system’s inputs to its outputs by describing the steps taken by the system in transforming inputs to outputs. That kind of description is precisely what cognitive scientists seek: a description of the internal causal sequences that bring about cognitive capacities, the latter understood as input–output relations. E.g., color-processing in early vision is modelled as an algorithm first summing responses from different types of cone, then weighting those sums, then adding and subtracting the weighted values, and eventually plotting the results in a three-dimensional space (Devalois & Devalois, 1993; Mancuso et al., 2010). That is a description, in terms of an algorithm, of the steps involved in turning a retinal input into a behavioral (or phenomenal) output—i.e. a description, in terms of an algorithm, of the process that brings about the capacity for color vision.

*Hierarchies* are descriptions of a process at different levels of abstraction, so that one level can be seen as implementing another. E.g., the processes involved in different capacities may rely, at a lower level of their hierarchies, on a small “set of standard (canonical) neural computations: combined and repeated across brain regions and modalities to apply similar operations to different problems” (Carandini, 2012; see also Carandini & Heeger, 2012). An understanding of this hierarchy does a number of things for cognitive science. Understanding the lowest level guides anatomical investigation into basic units and circuits, makes salient certain aspects of their causal structure, and guides the construction of models that make use of those units. Without this simplicity and structure, it would be prohibitively difficult to connect cognitive neuroscience to basic physiology and anatomy, or generally to lower levels of brain organization. An understanding of how lower levels compose higher levels also has benefits for modeling, making salient the relevantand practical levels of description for a model that explains a cognitive capacity (Yamins & DiCarlo, 2016). For all these purposes the benefits of computational formalisms are clear: their hierarchical properties are relatively well-understood; many computational formalisms are developed *for* their ability to compose complex functions from less complex ones; they are often developed to facilitate the use of a *small set* of functions to compose many complex ones—particularly relevant when we consider canonical computations as above; and they are developed to create and make intelligible complex relationships at different levels of grain and abstraction.

The importance of hierarchical and algorithmic explanation is the last point I’ll raise in explicit support of computational formalisms being a fruitful solution to (A). And we already saw that an algorithmic description *is* a theory about the process that brings about a cognitive capacity—that’s an important way that computational explanation achieves (B). Some of the preceding points about *salience* and the *guidance* of investigation (important features of scientific inquiry in general (Hookway, 2002)) also address (B) and (C). To carry this further, as I suggested above, computational formalisms often bring with them the conceptual schemes of their source disciplines. The assimilation of one system to into the conceptual scheme developed for another system is an important, widespread, and natural part of science (Dunbar, 2002; Nersessian, 2002), and conceptual schemes from the computing disciplines are useful ones in which to assimilate the brain. As I said, some of this follows straightforwardly from the discussion above, but there is more to say. E.g., considerations of *algorithmic complexity*—an important concept in computer science—drive discussions about the appropriateness of Bayesian models of the brain []. Considerations of computational efficiency (e.g., the “100 step rule”)—important in computer science and computer engineering—drove early debates in cognitive neuroscience (McClelland et al., 1986). It is because we think of the brain in computational terms that we investigate its canonical operations, as above. It is because we understand the properties of recurrent connections in neural networks that we look for recurrent connections among neurons—currently the subject of promising research. So assimilation into the conceptual schemes of computing disciplines provides many concrete benefits. If this seems to belabor the obvious, recall that the take-away is not that it is useful to think of the brain as a computer—that *is* obvious. The take-away is that we are making sense of this fact *without claiming that the brain is a computer.* All that the above requires is the use of computational formalisms to describe causal structure—no more metaphysically committal than the use of fluid dynamics to model a traffic jam. (The next paragraph will seem especially banal if this is not kept in mind.)

One more way that computational explanation serves the goals of computer science, perhaps falling under (B) and (C), is that it makes it possible, and relatively easy, to *build* the relevant models. Compare (Kriegeskorte & Douglas, 2018):“only synthesis in a computer simulation can reveal what the interaction of the proposed component mechanisms [of some theory] actually entails and whether it can account for the cognitive function in question” (Kriegeskorte & Douglas, 2018).[[20]](#footnote-20) It is a common refrain in the history of cognitive science that computational models make hypotheses specific and testable, and they do this partly by making them *buildable*, using our current technology (e.g. Churchland & Grush, 1999; Pylyshyn, 1984, p. xv; Sejnowski et al., 1988). To grasp the significance of this one need only imagine a theory on which the visual cortex is a kind of convolutional neural network, but imagine it proposed 50 years ago. The benefits this theory has because of *current* computing technology (benefits to do with prediction, ease of understanding, the possibility of proofs of concept, a familiarity with the model and an intuitive understanding of what it says about a target system, etc.) are the benefits I’m claiming computational models have in general.

[Review (Samuels, 2019, p. 107) on some of these general benefits.]

That’s all I’ll say about (A)–(C). The goal was to sketch a view of how computational explanation works and why it is so successful in cognitive science. The upshot is that *computational explanation works by using computational formalisms and the conceptual resources attendant on them to construct process models that capture the causal structures in the brain that bring about cognitive capacities*. If this is what computational explanation is, it requires no assumption that the brain is a computer. And we can see not only how computational explanation works, but why it is successful: it serves the explanatory needs identified in (A)–(C), in a way that serves the broader goals of cognitive science. And we can also see what makes computational explanation *distinctive* as a mode of causal explanation—the particular, powerful suite of tools, resources, and concepts it draws from.

Modeling *inference*? Considering neural activity as a type of inference? (Samuels, 2019, p. 108)

4 Objections

The bulk of the Pragmatic Approach is on the table. In this section I’ll consider some objections. Dealing with them will let me sharpen the Pragmatic Approach a little further. To bring out the most pressing worry, let’s start by returning to a version of the triviality problem.

*4.1 Empirical Content*

Though the triviality problem can’t arise in its original guise (it attacked the criteria for membership in the category computer, which are not invoked on my view), there is a way it might be resuscitated: if we don’t know what computational explanations say about their target systems, we haven’t fully understood how or why computational explanation works. I’ve understood computational explanation as a certain kind of modeling practice, but I haven’t said how to tell when a computational explanation *correctly* applies. That is, I haven’t placed any constraints on the empirical contentof computational explanations. And if there are no constraints on the empirical content of a model—on what it says about its target system—then why can’t we interpret it as saying whatever we like? Why can’t I give a computational process model of a rock as performing addition, and say that the model is correct as long as the rock runs through time-slices that correspond to the stages of the model’s addition algorithm? This was a long way to come, just to end up back at the original problem!

But the problem is only apparent. Computational explanations say something about the causal structure of their target systems, and how that structure brings about their capacities, and two constraints ensure that what they say about this structure is non-arbitrary.

First, note that since we’re considering computational explanations as models, this version of the triviality problem is a special case of a more general problem: what makes *any* model say what it says about the system it says it about? I don’t propose to answer this question here, but on the view above, the question of the content of a computational explanation is just an instance of this more general question of model reference (Frigg & Hartmann, 2018; Frigg & Nguyen, 2018). This is useful to note for two reasons. First, because scientific models work well in many cases, so we can be reasonably sure that it is not arbitrary what scientific models in general (and therefore computational explanations in particular) say about their target systems. And note that the problem of model reference is not or generally, solved by criteria for the target system’s membership in a particular category. The revision of the triviality problem I’m considering here would apply equally well to models of the solar system constructed in calculus equations, and the problem of why those models say what they say about their target systems is not solved by criteria for being a calculizer. Likewise for traffic and fluid dynamics, disinformation and virology, and so on. So the fact that we must confront the problem of model reference is no argument for the Metaphysical Approach. The assimilation of the triviality problem to the general problem of model reference is useful for a more practical reason too. Pressing this version of the triviality objection against only rival accounts of computation, or against computationalism itself, is incoherent. This version can only be pressed against every computational model, along with every scientific model, at once. That means it is a difficult objection to make consistently—probably prohibitively difficult for most philosophers.[[21]](#footnote-21)

So that’s one source of constraints on what computational theories say about their target systems: if you think there is a solution to the problem of model reference, you should not worry about arbitrariness in the empirical content of computational explanations. The second constraint comes from existing scientific knowledge. The goal of a computational explanation is to describe the causal structure in a system that brings about that system’s capacities. Not just any mapping from system to model is appropriate for this task. If a neuroscientist held that her model of memory was confirmed by connectome data because that data revealed that the brain had *just some* mapping to her model, she would be laughed out of the lab meeting. What counts as an appropriate mapping of model to brain (appropriate empirical content) depends on background neuroscientific knowledge, about (e.g.) which aspects of brain activity are involved in the tasks she’s modelling, which components of the brain are causally efficacious in the right ways, and so on. To explain a memory task, she might propose that *synaptic weights* correspond to certain terms in her computational model. This would be an appropriate assignment of content if the synaptic weights were understood to be causally implicated in memory in the required way, but it would be an inappropriate assignment if Gallistel and King (2009) were right that synaptic weights cannot be responsible for memory. The existence of empirical constraints on model interpretation is familiar to cognitive scientists—e.g., see discussions of “mappable” models (Yamins & DiCarlo, 2016) or “explanatory mechanisms” in model building (Blohm et al., 2020). For more concrete examples, see the debates over whether neural spike *rates* or spike *timings* are causally efficacious in the brain and should be the target of models, or similar debates between modeling population activity or modeling individual neurons and their connections (Barack & Krakauer, n.d.).A determination of the empirical content of computational models cannot be laid down by a philosophical theory. This does not mean it is left mysterious; it is just determined by more diverse and empirical considerations than one might have expected.

Aside from defusing the revised triviality problem, both these factors also give further content to the Pragmatic Approach. Computational explanation is successful as a general strategy because it helps meet (A)–(C), but *particular* computational explanations are successful insofar as their interpretations in terms of causal structure, which are determined partly by general considerations to do with scientific models and partly by background scientific knowledge about the brain, turn out to be true (or as true as the current context/idealization requires).

*4.2 Two More Versions of the Triviality Problem*

There are two less nuanced ways of re-introducing the triviality problem, both of which allow me to address some more general concerns. The recipe for a triviality problem is to take something we have or need criteria for, and to show that those criteria encompass too many entities. On the Pragmatic Approach there is nothing it is to be a computer, but there seems to be something it is to be a computational explanation, so, first, we can worry, *why isn’t every explanation (or why aren’t too many explanations) computational explanation?* And computational explanations are successfully applied only some of the time, or to only certain systems—there appear to be criteria for the *applicability* of computational explanations. So, second, we can wonder*, why isn’t computational explanation applicable to everything (or to too many things)?*

An answer to the first question is implicit in the account I’ve given: where formalisms and conceptual resources drawn from the computing disciplines help to meet explanatory needs (I’ve discussed (A)–(C), but there are surely others), then you have a computational explanation. Otherwise you don’t. Consider a model of a two-body system given in calculus equations. (A) does not appear to be met—though it presumably could be in a different model of the system. But even if it were, (B) is met only to a lesser degree, if at all—the conceptual resources required to understand and form hypotheses about the system do not come from a computing discipline. And (C) is not met at all: the heuristics and background knowledge required to understand the model and the system do not come from a computing discipline. So even if a two-body system were given a process model in a paradigmatic computational formalism, it would not be given a computational explanation.

We might worry about borderline cases—e.g., computer models of evolutionary processes that assume a sort of optimality to natural selection, and look for algorithms to achieve it.[[22]](#footnote-22) Say we have such a model, for which (B) and (C) are met to a large degree by conceptual resources from computing disciplines, but not to the same degree as in a typical computational model of (say) visual processing. In that case we should be happy to say the explanation is *closer to* a computational explanation, or *more of* a computational explanation, or is a more *paradigmatic* computational explanation. There is no reason to expect computational explanation to be a binary category, and since it is defined by the use of tools from computing disciplines, that use and those disciplines likely being fuzzily defined themselves, we should expect a fuzzy spectrum rather than strict criteria for counting as a computational explanation. This does not make it any harder to understand computational explanations or the source of their explanatory significance. That source was not their belonging to some strictly-defined category, it was their use of certain resources to meet certain needs. Some of those resources can be present—and therefore relevant to the explanations’ force—while others are not, or are only to various degrees.

I can now address an issue I postponed earlier. Some of the computational explanations I’ve mentioned, e.g. the ones to do with color vision, don’t use formalisms drawn from computing disciplines—they use arithmetic. Retinal ganglion cells are described as computing S–(L+M), e.g., where the letters refer to the responses of different cone types. They do, however, *conceptualize* the retina as following algorithms, one of the computational conceptual resources I pointed out above.[[23]](#footnote-23) And they may draw on knowledge from the computing disciplines, e.g., to do with the efficiency of different algorithms, or the use of population-coding. So there are more and less computational ways of describing color vision, and I don’t think it does any damage to the account if we allow that some only weakly counts as computational explanation. The explanatory role of these kinds of explanations can be captured partly by the account I’ve developed, and partly by more general accounts of non-computational explanation. There is no reason to demand that they be counted as full-blooded computational explanations when we have a good explanation of their function and success that doesn’t require them to be so counted.

What about the second question, above: why is it not *always* acceptable to use computational explanations, regardless of one’s target system or explananda? In one sense, it is! We should have no qualms with someone to whom computational formalisms and conceptual resources derived from computing disciplines are helpful for explaining (say) planetary systems or the weather, because accommodating this does not require a revisionary metaphysics according to which the planetary system *is* a computer. We should note, however, that computational explanations are in fact not helpful in most cases, at least to us in our current context. They do not help us make sense of the behavior of the weather, nor, e.g., rocks or walls or pails of water. The use of computational explanation should not be barred anywhere a priori, but in practice it is not appropriate in every case or context. We should perhaps expect computational explanation to be particularly successful for systems that have undergone a design process (including design by selection) to create a structure that efficiently generates appropriate outputs from inputs, because it is from disciplines creating and studying that kind of system that computational explanation draws most of its resources. So though there are no grounds to bar it a priori, a rock is unlikely to receive a successful or fruitful computational explanation. There remain further questions about the different explananda computational explanation might serve, along with deeper questions about the types of systems that will be most susceptible to it and how those systems could be identified. I’ll postpone these questions for further work, noting only that what I’ve said should be enough to dissolve the triviality worry I’m addressing.

*4.3 The Role of Causation*

I’ve described computational explanations as capturing certain kinds of causal structure in cognitive systems. Why not, then, return to the causal view of computation, according to which there is a criterion for performing a computation, and that criterion is that the computing system’s causal structure bears a certain relation to the algorithm or function computed? One problem is that, as I noted in Section 2, the causal view may not be restrictive enough to block triviality. E.g., Putnam claimed that his rock made all of its transitions causally (1991). I take it this point is still controversial (Chalmers, 1996).

But there is a more basic objection to the causal view that can be seen clearly now: it classifies every causal structure as computational, and thus misses what makes computational explanation sodistinctive and significant in cognitive science. And it does this precisely because it takes the Metaphysical Approach, focusing on criteria for computation. If computational explanations work because their target systems are computers, but being a computer is just having a causal structure, then computational explanations work in exactly the same way, and for exactly the same reasons, as every other causal explanation. On the Pragmatic Approach this is true, and important, in a general sense. But it is true and important *only* in a general sense. Computational explanations are causal explanations, but they are not *just* causal explanations. They have distinctive features that make them particularly suited to the task of cognitive science—features that are important to their explanatory role, some of which I set out above. The causal view, by focusing on criteria for computation, skirts all the interesting questions about computational explanation and its distinctive role in cognitive science. Though the Pragmatic Approach assimilates computational explanations to causal models in general, it makes room to explain what is so distinctive about them by focusing attention on the resources they bring to the modeling task.

*4.4 Naturalism and Objectivity*

There is a broadly-accepted desideratum that a theory of computation be naturalistic and objective. Here’s a formulation of naturalism: “The theory [of computation] should be naturalistic: it should not make the truth of implementation claims depend on human beliefs, interests, or values” (Sprevak, 2019, p. 177). And one of objectivity: “a good account of concrete computation should entail that there is a fact of the matter as to which computations are performed by which systems” (Piccinini, 2015, p. 12), where a *fact of the matter* is independent of anyone’s interpretation of or perspective on the systems (Piccinini, 2015, p. 11). This looks like a problem for the Pragmatic Approach. But, in fact, these desiderata must be accepted in conjunction with the Metaphysical Approach to support an objection to the version of the Pragmatic Approach, so the objection is circular. Why do we have these desiderata in the first place? For Piccinini, it is to ensure that our understanding of computation makes sense of cognitive science, where computational models are debated and justified, not just *freely assigned* (Piccinini, 2015, p. 11). But all along I have accepted strict constraints on applying and justifying computational models—the point of dropping the metaphysics of computation was to find new sources for precisely those constraints. Some of those sources are subjective: scientific goals, current states of knowledge, and so on. Insofar as one assumes that those goals, that knowledge, etc., have no place in an understanding of scientific explanation, one could reject the Pragmatic Approach. But that is an implausible assumption. An objection from the objectivity or naturalism desiderata is only plausible on the assumption that there is *something it is to be* a computer, with *that* *something* underwriting computational explanation. It is implausible that *that something*—the *what-it-is-to-be*, the essence, or nature, or criterion of computation—is observer-dependent (though see Rescorla, 2013). But on the Pragmatic Approach we aren’t talking about that something; we’re talking about a mode of explanation that is made sense of and justified by appeal to the goals it serves. And, as I’ve said, in that case the objectivity and naturalism desiderata are unnecessary and implausible, since their purpose (the restriction of the applicability and justification of computational explanation) is achieved by other means.

*4.5 What about the Computational Theory of Mind?*

The Pragmatic Approach to computational explanation has some potentially troubling implications for the philosophy of mind. Most importantly, the Metaphysical Approach helps to underwrite the Computational Theory of Mind (CTM). According to the CTM, to have a mind is to be a computer of a certain sort, and aspects of our mentality are explained by the kind of computers we are. Computational relations between beliefs and desires explain rationality, computational operations on sensory stimuli explain perception, and so on. If the CTM is true, then the mind literally is a physical computer, or is the computational structure of some physical system. And the best candidate physical system is the brain.

Perhaps this seems like a defeater: the Metaphysical Approach is implied by the CTM, and the CTM is our best theory of mind, so we had better not give it up, nor therefore had we better give up the Metaphysical Approach. The problem is that the CTM is taken to be our best theory of mind only because of—or, at least, it is supported primarily by—the deliverances of cognitive science. Specifically, by the success of computational explanation. But that is only support for the CTM *on* the Metaphysical Approach. If the success of computational explanations of the brain does not imply that the brain is a computer, then the success of computational explanation is no support for the claim that the mind is a computer either. So the CTM supports the Metaphysical Approach only circularly. An adherent of the CTM must find some other reason to accept the Metaphysical Approach. If they don’t, then the account I’ve given undermines a major source of support for the CTM. They may instead argue that the brain is a computer even though cognitive science does not support this claim. But in that case the CTM would support no objection to the Pragmatic Approach to computational explanation.

One more note on this. The problem for the CTM is not a problem for the *explanation* of the mind. Computational explanation still applies to mental capacities like reasoning, perception, and so on. But it does not underwrite a view about the *nature* or *metaphysics* of those mental capacities or the mind in general. It can’t help a philosopher argue that *what it is to be a reasoner* is to have a certain computational structure. It’s worth asking what we lose if we give up this latter project.

*4.6 The metaphysical appendix*

One way to hold on to a metaphysics of computation, while retaining all the benefits of the Pragmatic Approach, is to simply say that whatever systems receive legitimate computational explanations according to the Pragmatic Approach are *thereby* computers. That means that a system that meetsthe Pragmatic Approach’s criteria for the application of a computational explanation *thereby* meets the criteria for being a computer. So we get metaphysics through the back door—we give the account above, and add an appendix that says “and being explicable like thatis what it takes to be a computer”. This lets us classify the brain as a computer without requiring our metaphysics to do any heavy lifting, and I take it this is what Egan has in mind when she argues for a *metaphysically light* construal of computation (in conversation). Before I make what seems like a serious concession, let me note that this view—the Metaphysical Appendix View—accepts the claim that the metaphysics of computation are irrelevant to the understanding of computational explanation. The proponent of this view just has some other reason to want a metaphysics of computation (maybe it’s the kind of concept that *just can’t fail* to correspond to a category, even if the use of the concept relies not at all on the nature of that category). OK then, here is the concession: I have no problem with the appendix. Have your metaphysics, as long as you acknowledge that it plays no role in the understanding of computational explanation.

But note: harmless though they may seem, appendices are liable to burst, and an approach that insists on a metaphysical appendix leaves itself open to some troubling complications. The resulting metaphysics of computation would be “stancey” and observer-relative, as discussed above in connection with objectivity and naturalism. It is likely to be *graded* as well, given the discussion in section 4.2. These are not serious objections, because so far we have no reason to require a metaphysics of computation to be objective, observer-independent, etc. Perhaps a metaphysics that is relied upon in science should be all of those things, but one that exists only as an appendix cannot be held to these standards. One might as well criticize the category susie’s least favorite bubble gum for being subjective, indeterminate, etc. But those objections, misguided as they are, bring with them a dialectical context—the context of the question *What is it to be a computer?*—where they can be confused with good objections, or even grounds for rejection. I’ve taken the appendectomy because these complications are likely. It is better to reject the relevance of the metaphysics of computation to computational explanation *by* *rejecting that metaphysics itself*, because if you accept that there is a metaphysics of computation, just that it’s irrelevant to your questions, you will find yourself giving that metaphysics to an audience that can’t help but see it as relevant, and you will be pulled into a dialectical context that makes things far harder than they need to be.

5 Conclusion

Let me briefly summarize. Computational explanation in cognitive science works byusing formalisms drawn from the computing disciplines, and the conceptual resources attendant on them, to construct process models that capture the causal structures in the brain that bring about cognitive capacities. It is successful as a general strategy because it serves the needs of cognitive science, and it is successful in specific instances because (or if) it accurately describes the causal structures that bring about the behavior or capacity under investigation. On the view I’ve described, it is a non-trivial empirical matter whether a computational explanation is correct, and there are limits on which computational explanations appropriately apply to which systems, preserving the explanatory relationship between computations and the capacities they explain.

So far I’ve merely built up the Pragmatic Approach, letting the sense it makes of computational explanation stand as abductive evidence for it. And I’m content to have simply gotten this approach on the table, ready for further discussion and refinement. But in the introduction I promised you a further argument against the Metaphysical Approach. That argument is this: if there is a working account that does without the metaphysics of computation, then if you want an account that accepts such a metaphysics, *you need a reason to think you need it*. The necessity of this metaphysics is rarely argued for (though see footnote 18), but to endorse a view that requires a metaphysics of computation is to postulate a metaphysical entity—a *nature* of computation, a *kind* that physical things instantiate. An argument for the necessity of this assumption must specify some desiderata that the Pragmatic Approach doesn’t serve, but that the Metaphysical Approach (or the Metaphysical Appendix) does serve. Or it must give some other reason that we should expect a nature of computation to exist. Otherwise the Metaphysical Approach postulates metaphysical entities for no apparent gain—those entities are explanatorily redundant, ontologically superfluous, and contribute nothing to the understanding of computational explanation or anything else.

There is another way of coming at this point. To hold any specific Metaphysical Approach (the representational approach, the mechanistic approach, etc.), is not only to commit ontological superfluity, it is to understand cognitive science/scientists as committed to the existence of a natureof computation, and to a specific understanding of that nature. But these are commitments that cognitive scientists don’t intend to, or appear to, make themselves. To take just one example of how mainstream cognitive scientists see their practice, (Yamins & DiCarlo, 2016) understand their preferred type of computational explanation as a way of “formalizing knowledge about the brain’s anatomical and functional connectivity” so as to explain its cognitive capacities—not a metaphysical claim at all, and quite in line with the view I’ve defended here.

So, on the Pragmatic Approach we not only avoid making unnecessary metaphysical commitments, we also avoid unjustifiably attributing those commitments to cognitive science/scientists. This is just one count on which the Pragmatic Approach is preferable the Metaphysical Approach, but it reflects a broader range of considerations to do with the connection between cognitive science and philosophy. To build the bridges between these disciplines that we all seek, it will be important to avoid, as far as possible, foisting the assumptions and definitions of one field onto the other. The Pragmatic Approach avoids at least one such foisting that the Metaphysical Approach does not. And, in fact, my version of the Pragmatic Approach does so by focusing on the details of the context, practice, and function of computational explanation *for* cognitive scientists—something else that bridge-building requires, and that few accounts of computation do. So the Pragmatic Approach is solid ground for this kind of bridge-building.

To conclude, computational models provide a powerful, and crucial, lens on the brain—there’s a reason they’re (pretty much) the only game in town.[[24]](#footnote-24) Philosophers of cognitive science have been duly impressed by the computational lens, but have failed to see it *as* a lens, instead understanding computation as a property of the brain itself. This is a natural enough mistake—a good lens is not perceived; it is perceived through. But the oversight leads to errors: if we forget we’re looking through a lens, we are liable to believe that everything we see belongs to the brain itself. For that matter, philosophers of a certain sort are liable to leave the lens on, turn to a chunk of rock, and shudder to discover that *it* has all the brain’s computational properties too.[[25]](#footnote-25) When instead we understand computation as a lens, we truly begin to see what it does to its target, what it occludes and makes salient, what it adds, what it blurs, what it brings into focus, and how, in turn, it makes the brain intelligible as the organ responsible for the mind.

“What is distinctive of CCMs [Classical Computational Models] is that they characterize their *targets* as computational systems of a particular sort” (Samuels, 2019, p. 104).

References

Barack, D., & Krakauer, J. W. (n.d.). Two Views on the Cognitive Brain. *Draft*.

Blohm, G., Kording, K. P., & Schrater, P. R. (2020). A how-to-model guide for neuroscience. *ENeuro*, *7*(1), 1–12. https://doi.org/10.1523/ENEURO.0352-19.2019

Boden, M. (2006). *Mind as Machine: A History of Cognitive Science*. Oxford University Press.

Carandini, M. (2012). From circuits to behavior: a bridge too far? *Nature Neuroscience*, *15*(4), 507–509. https://doi.org/10.1038/nn.3043

Carandini, M., & Heeger, D. J. (2012). Normalization as a canonical neural computation. *Nature Reviews Neuroscience*, *13*(1), 51–62. https://doi.org/10.1038/nrn3136

Cetina, K. K. (1999). *Epistemic Cultures*. Harvard University Press.

Chalmers, D. J. (1996). Does a Rock Implement Every Finite-State Automaton? *Synthese*, *108*, 309–333.

Chalmers, D. J. (2011). A Computational Foundation for the Study of Cognition. *Journal of Cognitive Science*, *12*, 323–357.

Chirimuuta, M. (2014). Minimal models and canonical neural computations: the distinctness of computational explanation in neuroscience. *Synthese*, *191*(2), 127–153.

Churchland, P. S., & Grush, R. (1999). Computation and the Brain. In *The MIT Encyclopedia of the Cognitive Sciences* (pp. 155–157). MIT Press.

Cobb, M. (2020). *The Idea of the Brain*. Basic Books.

Craver, C. F. (2006). When mechanistic models explain. *Synthese*, *153*(3), 355–376. https://doi.org/10.1007/s11229-006-9097-x

Danks, D. (2019). Probabilistic Models. In M. Sprevak & M. Colombo (Eds.), *The Routledge Handbook of the Computational Mind* (pp. 149–158). Routledge.

Dennett, D. C. (1981). *Brainstorms*. MIT Press.

Devalois, R., & Devalois, K. (1993). A Multi-Stage Color Model. *Vision Research*, *33*(8), 1053–1065.

Dewhurst, J. (2018). Computing Mechanisms Without Proper Functions. *Minds and Machines*, *28*(3), 569–588. https://doi.org/10.1007/s11023-018-9474-5

Dougherty, J., & Callender, C. (2016). Black Hole Thermodynamics: More Than an Analogy? *Philosophy of Cosmology*. http://jamesowenweatherall.com/wp-content/uploads/2016/10/bht\_irvine.pdf

Dunbar, K. N. (2002). Understanding the role of cognition in science: the Science as Category framework. In P. Carruthers, S. Stich, & M. Siegal (Eds.), *The Cognitive Basis of Science* (pp. 154–171). Cambridge University Press.

Edsel, A. (2016). *Breaking Failure*. FT Press.

Egan, F. (1999). In Defence of Narrow Mindedness. *Mind & Language*, *14*(2), 177–194.

Egan, F. (2012). Metaphysics and Computational Cognitive Science: Let’s Not Let the Tail Wag the Dog. *Journal of Cognitive Science*, *13*(1), 39–49. https://doi.org/10.17791/jcs.2012.13.1.39

Egan, F. (2014). How to think about mental content. *Philosophical Studies*, *170*, 115–135.

Egan, F. (2019). The nature and function of content in computational models. In M. Sprevak & M. Colombo (Eds.), *The Routledge Handbook of the Computational Mind* (pp. 247–258). Routledge.

Fodor, J. A. (1968). *Psychological Explanation: An Introduction to the Philosophy of Psychology*. Random House.

Fodor, J. A. (1975). *The Language of Thought*. Harvard University Press.

Frigg, R., & Hartmann, S. (2018). Models in Science. In *Stanford Encyclopedia of Philosophy*.

Frigg, R., & Nguyen, J. (2018). Scientific Representation. In *Stanford Encyclopedia of Philosophy*.

Gallistel, C. R., & King, A. P. (2009). *Memory and the Computational Brain*. Wiley-Blackwell.

Grothe, B., Pecka, M., & McAlpine, D. (2010). Mechanisms of sound localization in mammals. *Physiological Reviews*, *90*(3), 983–1012. https://doi.org/10.1152/physrev.00026.2009

Ham, Y.-G., Kim, J.-H., & Luo, J.-J. (2019). Deep learning for multi-year ENSO forecasts. *Nature*, *573*(7775), 568–572. https://doi.org/10.1038/s41586-019-1559-7

Headington, M. R., & Riley, D. D. (1997). *Data Abstraction and Structures Using C++*. Jones and Bartlett.

Hillis, D. W. (1998). *The Pattern on the Stone: The Simple Ideas that Make Computers Work*. Basic Books.

Hookway, C. (2002). Emotions and epistemic evaluations. In P. Carruthers, S. Stich, & M. Siegal (Eds.), *The Cognitive Basis of Science* (pp. 251–262). Cambridge University Press.

Jacobs, G. H. (2014). The discovery of spectral opponency in visual systems and its impact on understanding the neurobiology of color vision. *Journal of the History of the Neurosciences*, *23*(3), 287–314. https://doi.org/10.1080/0964704X.2014.896662

Kriegeskorte, N., & Douglas, P. K. (2018). Cognitive computational neuroscience. *Nature Neuroscience*, *21*(9), 1148–1160. https://doi.org/10.1038/s41593-018-0210-5

Lake, B. M., Ullman, T. D., Tenenbaum, J. B., & Gershman, S. J. (2017). Building machines that learn and think like people. *Behavioral and Brain Sciences*, *40*. https://doi.org/10.1017/S0140525X16001837

Lazebnik, Y. (2004). Can a biologist fix a radio? - Or, what I learned while studying apoptosis. *Biochemistry (Moscow)*, *69*(12), 1403–1406. https://doi.org/10.1007/s10541-005-0088-1

Lieberman, H. (2006). The continuing quest for abstraction. *Lecture Notes in Computer Science*, *4067 LNCS*, 192–197. https://doi.org/10.1007/11785477\_12

Lloyd, S., & Ng, Y. J. (2004). Black Hole Computers. *Scientific American*, *291*(5), 52–61.

Machamer, P., Darden, L., & Craver, C. F. (2000). Thinking about Mechanisms. *Philosophy of Science*, *67*(1), 1–25.

Mancuso, K., Mauck, M. C., Kuchenbecker, J. A., Neitz, M., & Neitz, J. (2010). A Multi-Stage Color Model Revisited: Implications for a Gene Therapy Cure for Red-Green Colorblindness. *Advances in Experimental Medicine and Biology*, *664*, 631–638.

Marr, D. (1982). *Vision*. W.H. Freeman and Company.

Matthews, R. J., & Dresner, E. (2017). Measurement and Computational Skepticism. *Nous*, *51*(4), 832–854. https://doi.org/10.1111/nous.12142

McClelland, J. L., Rumelhart, D. E., & Hinton, G. E. (1986). The Appeal of Parallel Distributed Processing. In D. E. Rumelhart, J. L. McClelland, & P. R. G. The (Eds.), *Parallel Distributed Processing* (pp. 3–44). MIT Press.

McCulloch, W. S., & Pitts, W. (1943). A logical calculus of the ideas immanent in nervous activity. *The Bulletin of Mathematical Biophysics*, *5*(4), 115–133. https://doi.org/10.1007/BF02478259

Milkowski, M. (2013). *Explaining the Computational Mind*. MIT Press.

Miller, M. (2014). *Minds Online: Teaching Effectively with Technology*. Harvard University Press.

Nersessian, N. J. (2002). The cognitive basis of model-based reasoning in science. In P. Carruthers, S. Stich, & M. Siegal (Eds.), *The Cognitive Basis of Science* (pp. 133–153). Cambridge University Press.

Neumann, J. von. (1958). *The Computer and the Brain*. Yale University Press.

Nityananda, V., & Read, J. C. A. (2017). Stereopsis in animals: Evolution, function and mechanisms. *Journal of Experimental Biology*, *220*(14), 2502–2512. https://doi.org/10.1242/jeb.143883

Peacocke, C. (1994). Content, Computation and Externalism. *Mind & Language*, *9*(3), 303–335.

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

Putnam, H. (1991). *Representation and Reality*. MIT Press.

Pylyshyn, Z. W. (1984). *Computation and Cognition*. 1984.

Pylyshyn, Z. W. (1993). Computing in Cognitive Science. In M. I. Posner (Ed.), *Foundations of Cognitive Science* (pp. 49–92). MIT Press.

Rescorla, M. (2013). Against Structuralist Theories of Computational Implementation. *The British Journal for the Philosophy of Science*, *64*, 681–707. https://doi.org/10.1093/bjps/axs017

Samuels, R. (2019). Classical Computational Models. In M. Sprevak & M. Colombo (Eds.), *The Routledge Handbook of the Computational Mind* (pp. 103–119). Routledge.

Scheutz, M. (2012). What it is not to Implement a Computation: A Critical Analysis of Chalmers’ Notion of Implemention. *Journal of Cognitive Science*, *13*(1), 75–106. https://doi.org/10.17791/jcs.2012.13.1.75

Sejnowski, T. J., Koch, C., & Churchland, P. S. (1988). Computational Neuroscience. *Science*, *241*(4871), 1299–1306.

Shadmehr, R., & Wise, S. (2005). *The Computational Neurobiology of Reaching and Pointing*. MIT Press.

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

Shagrir, O. (2006). Why We View the Brain as a Computer. *Synthese*, *153*(3), 393–416. https://doi.org/10.1007/sll229-006-9099-8

Shagrir, O. (2018). In defense of the semantic view of computation. *Synthese*, *January*. https://doi.org/10.1007/s11229-018-01921-z

Shea, N. (2013). Naturalising Representational Content. *Philosophy Com*, *8*(5), 496–509.

Simon, H. A., & Newell, A. (1973). Human Problem Solving: The State of the Theory in 1970. *American Psychologist*, *26*(2), 145–159. https://doi.org/10.2307/2063712

Smith, B. C. (1999). Computation. In R. A. Wilson & F. C. Keil (Eds.), *The MIT Encyclopedia of the Cognitive Sciences* (pp. 153–155). MIT Press.

Sprevak, M. (2019). Triviality arguments about computational implementation. In *The Routledge Handbook of the Computational Mind* (pp. 175–191). Routledge.

Sun, R. (2008). Introduction to Computational Cognitive Modeling. In R. Sun (Ed.), *The Cambridge Handbook of Computaitonal Psychology* (pp. 3–20). Cambridge University Press.

Thagard, P. (2002). The passionate scientist: emotion in scientific cognition. In P. Carruthers, S. Stich, & M. Siegal (Eds.), *The Cognitive Basis of Science* (pp. 235–250). Cambridge University Press.

Thoreson, W. B., & Dacey, D. M. (2019). Diverse Cell Types, Circuits, and Mechanisms For Color Vision in The Vertebrate Retina. *Physiol Rev*, *99*, 1527–1573.

Yamins, D. L. K., & DiCarlo, J. J. (2016). Using goal-driven deep learning models to understand sensory cortex. *Nature Neuroscience*, *19*(3), 356–365. https://doi.org/10.1038/nn.4244

1. Arguments for the computational approach in general (as opposed to arguments for specific computational explanations) are also common (e.g. Fodor, 1975; Gallistel & King, 2009; Pylyshyn, 1984, 1993). [↑](#footnote-ref-1)
2. This question is often posed in terms of the *justification*, *foundations*, or *legitimacy* of computational explanation (e.g. Chalmers, 2011; Dennett, 1981, p. xvii). But the question, *why is it successful as a mode of explanation*, can conveniently subsume these. (Thanks to [redacted for blind review] for urging the simpler framing.) [↑](#footnote-ref-2)
3. The research here, as elsewhere in philosophy, is modeled on a very old style. When Socrates asks Euthryphro why it is correct to say that taking one’s father to court is pious, it is only the beginning of an answer for Euthyphro to say that the act can instantiate the property of piety. The question is about that property itself. [↑](#footnote-ref-3)
4. For a representative example of this approach, see (Chalmers, 2011, p. 325): “We cannot justify the foundational role of computation without first answering the question: *What are the conditions under which a physical system implements a given computation?*” (his italics). Another example is (Sprevak, 2019): “A theory of implementation tells us which conditions the physical system needs to satisfy for it to implement the computation.” Non-criterial approaches are also possible: perhaps being a computer is a matter of similarity to a paradigm, or family resemblance. But since these approaches are not generally taken, I’ll set them aside. [↑](#footnote-ref-4)
5. This is not the only use to which computational models are put. They can be more than phenomenal models, but still less than how-explanations. E.g., they can specify the *optimal* functioning of a system [Gomez], or give a *why-explanation* for certain facts about it (Chirimuuta, 2014). The point is well taken, but these uses of computational explanation are not my target here. [↑](#footnote-ref-5)
6. Egan attempts to escape these problems criticism by combining the Metaphysical Approach with a *metaphysically light* understanding of computation (Egan, in correspondence). I’ll explore this in section 5. [↑](#footnote-ref-6)
7. We might say the brain is performing every computation, but for some other reason only some are explanatorily relevant. But then we have to say what makes one computation explanatorily relevant, and this reframing offers no extra traction on the questions I’m considering here. [↑](#footnote-ref-7)
8. The causal view also appears unable to account for the apparent *environmental* individuation of computational processes (see Shagrir, 2001; [redacted for blind review] forthcoming a; Shea, 2013). [↑](#footnote-ref-8)
9. See (Chalmers, 2011, p. 334) and (Sprevak, 2019, p. 177) ﻿on the need for computation to be grounded in well-understood notions. [↑](#footnote-ref-9)
10. Marr (1982) expresses a similar sentiment (p. 23). Also see (Fodor, 1968; Kriegeskorte & Douglas, 2018; Sun, 2008) on computational models as process models. [↑](#footnote-ref-10)
11. The following also holds for less traditionally “cognitive” capacities, like temperature control or emotion regulation. [↑](#footnote-ref-11)
12. For a detailed history of this approach, as the attempt to understand the mind as a machine, see (Boden, 2006). For a less detailed history, see (Cobb, 2020). [↑](#footnote-ref-12)
13. An important caveat is that we do not want the most detailed, or even the most accurate, model of causal structure. We want a model that coarse-grains, and idealizes, and is occasionally outright inaccurate, wherever those features are theoretically fruitful. E.g., we want models that coarse-grain so as to explain the successful behavior of an organism, even if that means they leave out some the ways that behavior can be unsuccessful. We would model a simple calculator with an addition-function, not an addition-except-where-the-calculator-errs function. So in the case of computational explanation, like most other cases of causal modeling, accuracy with respect to causal structure is just one goal, tempered by others. I’ll set this aside for now, but see ([redacted for blind review] forthcoming), where I discuss the role of *representations* in computational models, providing exactly this sort of idealization. [↑](#footnote-ref-13)
14. Talk of organization and processes sounds like New Mechanism (Machamer et al., 2000), but I intend what I say to be neutral on the kind of causal structures identified in computational explanation. [↑](#footnote-ref-14)
15. This is not all that a formalism and its associated conceptual resources should do. E.g., it may be important to frame *simple* theories, or ones that appeal to our sense of elegance (Thagard, 2002). But I’ll focus on (A)–(C). [↑](#footnote-ref-15)
16. Also, to the extent that artificial intelligence informs cognitive science, it will inform the kinds of computational formalisms that cognitive science uses (Kriegeskorte & Douglas, 2018). Since we cannot limit in advance the formalisms that artificial intelligence uses, we cannot limit the formalisms that flow into computational cognitive science. [↑](#footnote-ref-16)
17. Does there have to be *something it is to be* a computer to make sense of the notion of a computing discipline? No—disciplines are rarely defined by criterial conditions for their subject matter. (According to what criteria are Judith Butler and Saul Kripke studying the same subject matter, distinct from other disciplines?) If we want to know what makes something a computing discipline, we should instead look to the social, technological, and intellectual significance of the artefacts or systems they study, the sociology and politics of disciplinary division (Cetina, 1999, Chapter 1), and their overlapping concepts, background knowledge, and methods.

    It is also sometimes claimed that one or another computing discipline itself presupposes criteria for computation. E.g., “The computational sciences already (explicitly and implicitly) classify physical systems into those that compute and those that do not” (Sprevak, 2019, p. 176). But I see no reason to think those sciences *classify* those systems, rather than just applying computational notions only where they are useful. I don’t know of any thesis in computer science, e.g., that depends on a claim that one physical system *really is* a computer and another isn’t. And if they do, it is a further step to claim that “really being” a computer in this sense means falling into a category *that is also significant for the purposes of cognitive science*. (If the category is not significant for the purposes of cognitive science, it makes no difference to the argument here.) Note that the issue is not about criteria for *abstract* systems to be computational—computer science arguably has that in the Church-Turing Thesis. What computer science does not have are criteria for physicalsystems to implement the abstract ones. [↑](#footnote-ref-17)
18. I will return, in the next section, to borderline cases that may not be captured by my definition, like the arithmetical operations mentioned above. [↑](#footnote-ref-18)
19. The examples here are simplistic for the sake of explanation. Functional abstraction is most commonly discussed in more demanding contexts, e.g. the construction of an abstraction method to manage complex databases (Lieberman, 2006), or the construction of programming languages that can specify complex functions as an abstraction of the machine language run on a computer’s hardware (Headington & Riley, 1997). [↑](#footnote-ref-19)
20. A computer simulation is understood, in this context, as a process model. [↑](#footnote-ref-20)
21. Others have made similar points. Matthews & Dresner (2017) argue that triviality arguments against accounts of computation have the same structure as triviality arguments against *any* attribution of numerical properties to physical systems, and so cannot hold. The advantage of the Pragmatic Approach is that it can also say something positive about computational explanations and the reasons for their remarkable success. [↑](#footnote-ref-21)
22. Another example, to which all of the following will apply, is the use of computational notions in physics, e.g. in the information-theoretic notion of entropy applied to black holes (Dougherty & Callender, 2016; Lloyd & Ng, 2004). [↑](#footnote-ref-22)
23. Another example: Kosslyn’s use of the concept of a Visual Display Unit, like in commercial computers, to explain visual imagery—but without an associated description in a computational formalism. [↑](#footnote-ref-23)
24. Here “computation” doesn’t mean formal symbol manipulation, as it did when Fodor made the *only game in town* argument (1975), but the broader notion defined in section 3. [↑](#footnote-ref-24)
25. As in (Chalmers, 1996; Milkowski, 2013; Piccinini, 2015), and so on. [↑](#footnote-ref-25)