Computing in the nick of time

J. Brendan Ritchie[1†] and Colin Klein[2]
[1] National Institute of Mental Health
[2]The Australian National University
[†] j.brendan.w.ritchie@gmail.com

1 Introduction

We tend to think of computers as in the business of generating answers to questions. If a device is engineered to perform addition, it has been well-designed just in case it generates the appropriate sum given the input values. Just as addition is defined abstractly, the correctness of the sequence of steps the device carries out can also be characterized abstractly, in a manner independent of the particular medium in which they are implemented. This medium-independence of computational descriptions has shaped common conceptions of computational *explanation*. So long as our goal is to explain how a system successfully carries out its computations, then we only need to describe the abstract series of operations that achieve the desired input-output mapping, however they may be implemented. Put simply, since the *explanandum* is medium-independent so too is the *explanans*.[1]

One might wonder whether all computational explanation works like this. We argue that it does not. At least, not when the *explananda* essentially involve the *time* at which certain events must take place. We begin with an example, spell out its consequences for how we think about computational explanation, consider some objections, and then sketch some further implications for how we think about computational implementation as well as explanation in cognitive science.

[1]The proponents of this view are legion. Classic defenses include Fodor (1975); Cummins (1989); Newell (1990); Pylyshyn (1984); Simon (1992). More recent defenses include (among others): Chalmers (2011); Chirimuuta (2014, 2020); Coelho Mollo (2018); Gallistel and King (2011); Weiskopf (2011).

2 Pac-Man

The Atari 2600 port of Pac-Man was a commercial success, but it was critically panned due to visual and gameplay differences from the original arcade version. These differences can be explained by the vastly constrained computing resources available to Atari programmers, and the interaction between these resources and the chips that drove the display.

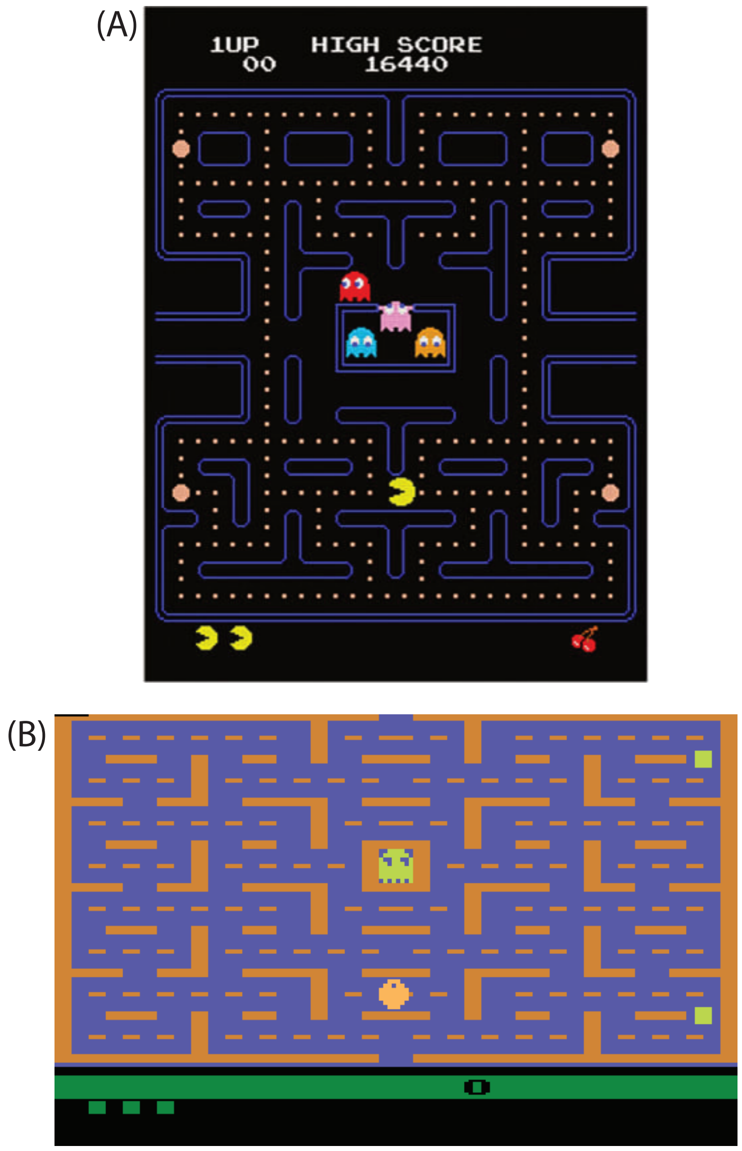

Figure 1: Playfields for two implementations of Pac-Man. (A) The original arcade version. (B) The Atari 2600 port.

Consider, for example, why the 'wafers' munched by Pac-Man were rectangles in the Atari version rather than round dots as in the original. Unlike the arcade hardware, the 2600 lacked dedicated video RAM. Hence the port had to render wafers as part of the background, using the same mechanism that renders the walls and border (Figure 1). But the Television Adaptor Interface (TIA) chip at the heart of Atari's graphics system was designed to render horizontally symmetric playfields. As wafers were removed asymmetrically, the playfield had to be updated in real time, while being drawn, and while players navigated the maze. To address this engineering problem, Montfort and Bogost explain that:

> . . . the program must first set the playfield register graphics for the left half of the screen during horizontal blank. Then, as the electron beam passes across the screen, it must change those registers just before the second half of the screen starts. This technique requires careful processor timing as well as additional RAM storage for the state of each pellet. Worse yet, the positions for each remaining pellet need to be translated from data in RAM into the unique display requirements of the TIA playfield, which does not simply write its two and a half bytes in consecutive, high-to-low bit order. To get the dots on the screen, the program tracks their states separately from their positions on-screen, performing a series of computationally expensive bitwise operations to install the pellet data into the maze playfield locations, which in turn use up valuable RAM. Maze and pellet logic — relatively simple for the arcade cabinet, given its hardware affordances — were very challenging on the Atari VCS.(Montfort and Bogost, 2009, p.69)

The requirement of a rapid redraw deadline of the background is a result of multiple real-time constraints. First, the MOS 6502 used as the CPU had a fixed, relatively slow clock speed. Second the NTSC television standard for cathode-ray televisions specifies a strict and relentless pace for the beam, rendering one line every $63.6\mu s$. Failure to meet these constraints would mean errors sufficient to render the game unplayable. So, given the speed of the CPU, the computational constraints of the TIA, the demands of cathode-ray televisions, and the primitive operations available from the CPU itself, the wafers could not be rendered rapidly enough to make them symmetrical: the best programmers could only make elongated lozenges.

Step back from the details and notice the explanatory strategy at work. It seamlessly mixes the sequencing of operations (such as the primitive bitwise operations available on the 6502) with real-time considerations. The timing of the arcade and Atari implementations of the game Pac-Man is more or less equivalent in terms of the game and how it is played, even if the playfield in the latter case leaves much to be desire aesthetically. Yet there is no way to explain *how* this timing is achieved without attention to the hardware of the platforms. Conversely, attention to timing in turn explains why various other computational decisions were made—why the system was programmed in the way that it was, why it has (or lacks) certain capacities, and so on. These are all familiar sorts of computational explanations: they say why one sequence is the case rather than another. Yet they all incorporate timing as well as sequencing.

3 Explaining in real-time

The CPU of the Atari is an example of an "embedded" or "cyber-physical" system: it is a device that performs computations in order to interface with other things that may or may not compute, such as mechanical or electronic devices that interact with the world (Lee, 2008). Embedded systems are paradigmatic examples of *real-time computing*, which is sensitive to the absolute time at which operations take place, not just the sequencing of those operations, (Lee, 2018; Shin and Ramanathan, 1994; Stankovic, 1988). In other words, when it comes to real-time computing, correct timing is *constitutive* of the computing task being performed. This point is usually expressed in terms of the presence of *deadlines*, which are classified into three types. A "hard" deadline is one where a failure to meet it is a total system failure; a "firm" deadline is one where sometimes missing it is acceptable, but results past a deadline have no use; and a "soft" deadline is one where the usefulness of a result degrades after the deadline. The Atari 2600 is an excellent example of a real-time computing system with hard deadlines: regular failure to meet them means game-play is ruined. What then explains its success?

This question has a complex answer.[2] But as we touched on above, broadly speaking it includes details about many timing requirements, which account for the clever hardware and software engineering of developers. Abstract characterizations of the rules of Pac-Man, or the program used for running the game, are part of the explanation, to be sure. The explanation remains *computational*: it has an ineliminable core that details the relevant computational operations and their results. But such abstract characterizations are not enough. Other programs, which could also be implemented on the MOS 6502 CPU might generate the correct outputs given the game state inputs, but not quickly enough to coincide with the redrawing of the playfield. Nor can focusing solely on the abstract computations being performed account for many features of the port that are a byproduct of the hard deadlines posed by the Atari's hardware and cathode-ray televisions. Limiting our explanation to such a characterization leaves opaque all the differences between the original arcade version of the game and the Atari port. For example, we explain why Pac-Man wafers have the shape they do by appeal to the computational aspects of drawing wafers *and* the how the timing of these operations interacted with the TIA.

To summarize, ignoring timing results in explanations of how the Atari 2600 meets its deadlines that are at best partial and superficial, and at worst no explanation at all. Mere similarity of input-output is not enough to distinguish sequences which compute what we care about from those that do not.

We belabor these points because the scope of computational explanation is beholden to how we characterize the successful performance of computing tasks. Many accounts implicitly or explicitly assume the following: computing successfully is solely a matter of generating

[2] Indeed it is for this reason that Montfort and Bogost (2009) wrote a whole book on the topic.

the appropriate outputs, given the inputs.[3] This input-output mapping, and any formal procedure that achieves this mapping, are wholly abstract; they are defined without any particular hardware details in mind. Yet, describing the procedure still affords an explanation of how the task is achieved. So the abstract characterization of computing tasks motivates the characterization of a computational explanation as the description of the abstract procedure for accomplishing the task. The *explanans* is abstract because it is assumed the *explanandum* is as well.

What exactly counts as the "abstract" computational description is of course up for debate. However, the same point applies whether abstract computation is characterized in terms descriptions of symbols and the rules defined over them (Fodor, 1975; Pylyshyn, 1984); organizationally invariant properties of a system (Chalmers, 2011); or the relationship between variables in mathematical models that map onto a physical system at some level of abstraction (Chirimuuta, 2014, 2020). In each case, equating computational explanation with the description of the relevant abstraction presupposes that the computing task of interest is similarly abstract.

However, a focus on abstract computation is ill-suited for describing the computing success of embedded, real-time computing systems. The correct of the output for such systems is partially determined by their deadlines, which are *defined* based on the other components a system interfaces with and its own hardware limitations. Since meeting these deadlines are not independent of how computations are implemented, one cannot explain how they are met without taking account of the nature of the interface and the hardware of the system. Hence, because the computing task cannot be described in solely formal terms, neither can an explanation of how the task is performed. What, then, does an account of computational explanation look like, if real-time computing is used as a starting point? We believe that existing approaches are suggestive. Here we will briefly consider one of them.

According to the *mechanistic* view computational explanation involves positing computing mechanisms: sets of organized, causally-related components that have the (teleological) function of processing medium-independent vehicles in accordance with rules defined over the vehicles (Kaplan, 2011; Milkowski, 2013; Piccinini, 2007, 2015). But whether a physical system has this function will depends on the larger context in which its embedded, as some proponents of the view recognize (Piccinini, 2007, 219-220, Piccinini, 2015, 138-139).[4] This fits nicely with our picture of real-time computing since the broader context, of how a device interfaces with its environment, determines the deadlines that are constitutive of the computing tasks it is carrying out. The mechanistic approach may not be the only one that has the resources to describe real-time computing. But any such view will have to recognize the crucial roll of embedding context in determining the scope of computational explanation (Fresco, 2021; Harbecke and Shagrir, 2019; Lee, 2020).[5]

[3]For a list of those committed to this view, see footnote 1.

[4]Though other proponents disagree, maintaining that what computations a system computes are determined solely by its intrinsic properties (Dewhurst, 2018; Coelho Mollo, 2018). This version of the mechanistic view is obviously ill-fit to describing real-time computing (cf. Harbecke and Shagrir, 2019).

[5]Such a mechanistic view, or other some contextually-sensitive view of computational explanation, would

The foregoing suggests that there is also no problem of indeterminacy when it comes to explaining real-time computing. The issue is supposed to arise from the apparent tension between two claims: on the one hand, that computational explanation requires a privileged computational description at a time; and on the other, any physical system may implement many computations at the same time. For example, in isolation, the very same system may be describable as implementing both an OR and an AND gate (Shagrir, 2001, 2020). Many solutions have been proposed to this problem, including ones that emphasize the importance of the context in which computing devices are embedded (for overview of the available views, see Curtis-Trudel, 2022; Papayannopoulos et al., 2022). From our vantage point, however, the problem is an artefact of starting-off with a conception of computing tasks that eschews matters of timing, which is inherently determined by the context in which a device is embedded. When deadlines are involved, the operations either causally contribute to the system meeting its deadlines for interfacing with other devices, or they do not. There is no ambiguity about which operations implemented by the CPU of the Atari 2600 contribute to the redrawing of the playfield, for example.[6]

More generally our argument can be restated in accordance with the influential information-processing framework of Marr (1982). Proponents of more abstract characterizations of computational explanation are quick to emphasize the importance of what Marr called the "computational theory", which specifies what function is computed, along with the algorithms that realize the function, as essential and distinct from the implementation details. However, for present purposes the most critical aspect of the computational theory for Marr is a specification of *why* the function is computed (Ritchie, 2019; Shagrir, 2010). To use Marr's example, the requirements of financial transaction accounts for why a cash register performs addition, as opposed to some other mathematical operation. But the "why"-component of the computational theory looks very different when sums are being calculated by the autoland system of a commercial jet rather than a point of service. For real-time computing, timing is inherent to characterizing why an operation is performed, and so it is an ineliminable component to any explanation of how the computing task is performed.

4 Objections and replies

For all we have said, the instinct that computational explanation *as such* only concerns abstract computations may persist. There are a number of objections that might be used to codify this conviction. None of them work.

have to be modified, however, if computation is not medium-independent (as discussed below). In the case of the mechanistic view, this would be simple to accommodate since computing mechanisms are a special case of functional mechanisms more broadly and instances of real-time computing will straightforwardly already qualify as mechanistic in this broader sense.

[6]Strictly speaking there are may be two sense of computational indeterminacy (Papayannopoulos et al., 2022): (i) how physical states of a device are grouped together relative to abstract states; and (ii) how the abstract states are interpreted to determine what function the device carries out. We believe neither of them present special problems in the case of real-time computing.

Objection: Real-time computing is an unusual, non-standard case.
Reply: While only a few decades ago real-time computing was characterized as a "new" discipline in computer science and engineering (Shin and Ramanathan, 1994), the increased presence of computing devices in our everyday life is largely due to the a proliferation of real-time computation (Buttazzo, 2011). Furthermore computations that require inter-process coordination with other computers (such as internet routers) also have real-time aspects: correct computation depends not just on the ordering of the computational steps themselves but in their synchronization with other, distinct computational systems (Klein, 2020). Hence, real-time computing is in fact ubiquitous.

Objection: Timing is already recognized as a "constraint" on engineering computers but is the sort of constraint that is relevant only to the implementation details rather than computational explanation as such.
Reply: Since deadlines are part of what *define* the relevant computing tasks for real-time systems, timing is not merely a constraint on their explanation – nor can it be relegated to the implementation details. For example, the slow clock speed of the MOS 6502 microprocessor and the fixed timing of the NTSC standard are constraints that help determine the hard deadlines for the Atari port of Pac-Man. But the resulting deadlines are part of the *explananda*. If the timing needed to meet these deadlines is not accounted for, the phenomenon has not been explained in the first place.

Objection: The important feature of computational explanation is that it describes the rules that govern the operation of a system, which allows us to generalize across implementations. For example, the rules of Pac-Man are the same for both the arcade original and Atari port.
Reply: first, this confuses *explanandum* and *explanans*. Discovering that the behavior of a device conforms to the rules of addition is not a computational description of how it maps inputs to outputs. Second, it risks begging the question, since it leaves implicit that part of the 'rules' that govern Pac-Man are specific to it being a *video* game, and those have to do with timing such as the latency between a player initiating movement of the avatar the accompanying change in the playfield.

Objection: One still needs an abstract description of the operations carried out by the Atari port of Pac-Man as part of the explanation.
Reply: Agreed. However, crucially in cases of real-time computing such a description is a necessary, but not sufficient, component of the explanation because of the presence of computing deadlines. Abstract computation is still core to the explanation when it comes to real-time systems, but so are the implementation details that account for how the relevant deadlines are met.

Objection: Timing is a feature of computational complexity, which is characterized in abstract, medium-independent terms.
Reply: The computational complexity of an algorithm is not related to time as such, but rather how resources scale with some measure n of the problem, and whether or not this scaling of the number of steps it takes to solve the problem is *efficient* (Aaronson, 2013). In contrast, real-time computing is not a matter of how many steps are taken to solve a

problem but rather about the *predictability* of the timing of computations carried out by a device and the relationship of those computations to other relevant external features of the world (Lee, 2018; Stankovic, 1988).[7]

Objection: Real-time computing is just computing done quickly.
Reply: It is not (Lee, 2018; Stankovic, 1988). All things being equal, it may be beneficial for a computing device to generate solutions to problems more quickly.[8] Conversely, the Atari 2600 is interesting precisely because it has a fairly slow CPU and because the fixed timing of the NTSC standard means that it is possible to be too *fast* as well as too slow. It is accurate timing, not mere speed, which counts.

Objection: We can always take the computational problem of computing A and turn it into the problem of computing (A, T), where T is the time at which A ought to be given. But this is still an atemporal problem, showing that computation itself remains timeless.
Reply: Real-time computing is more than just operating relative to a bounded execution time T; it is about predictability (Lee, 2018). Furthermore, knowing that something ought to be done at T is not the same as explaining how a system predictably carries out the operation at T, and it is the latter that is the *explanandum* when it comes to real-time computing.

Objection: The argument overgeneralizes. If time is important to computational explanation, why not cost, or size, or aesthetics, or any of the other engineering constraints that real-world systems operate under (cf. Pylyshyn, 1979; Weiskopf, 2004)?
Reply: If real-time computing was simply a matter of computing quickly (or slowly), then our argument would indeed overgeneralize, since virtually any design requirement could be considered part of the "computational" description of a device. However, as we have emphasized, timing is considered constitutive of the task being performed because correctness of an operation is relative to meeting certain deadlines. These deadlines demand accurate, predictable operation for interfacing with other systems. They are not simply bounds on operation time.

These other features of real-world systems would only relate to the explananda of computational explanation if they can similarly be shown to determine the correctness of the computing task being performed. However, the case is not easily made. We might require an adding machine to be built to be beautiful and cheap—but if the result is hideous and expensive, the device does not obviously cease to function correctly. In contrast, it is easy to imagine devices that are pleasing to the eye and engineered with cheap parts, which do not operate to the desired level of precision (as many online reviewers of electronic devices can attest). However, this further illustrates how these other engineering constraints and real-time computing demands come apart. Absent compelling cases to the contrary, we conclude

[7]Thus our claim is not that computational explanations must be tractable. Though plausibly they should be (Van Rooij, 2008).

[8]In scientific computing actual speed to publication matters quite a bit for success, creating what Hooker (2021) calls a "hardware lottery." But in such cases speed is considered a *performance metric*, not a feature of the problem that needs to be solved.

there is no threat of overgeneralization.[9]

5 Implications

Examples of real-time computing point to a conception of computational explanation that inherently involves considerations of real-time duration. This has important implications for how we think about computational implementation, and also the form of computational explanations in cognitive science.

5.1 Computational implementation

Many traditional problems for theories of implementation have arisen in the context of considering timeless, highly mathematized computational explanations. Thinking about real-time computation can shed light on the special assumptions needed to generate these problems, and where those assumptions might be reconsidered.

For example, an account of implementation must be extensionally adequate, delivering the correct verdict on paradigmatic cases of when physical systems carry out computations and resisting standard problem cases (Piccinini, 2015; Ritchie and Piccinini, 2018). A main challenge to achieving such adequacy is the threat of pancomputationalism, the view that all physical systems compute some function.[10] Yet we note that pancomputationalism is a much less pressing challenge for real-time computing. Whether a rock instantiates every finite state automaton, it cannot take the places of an autoland system on a plane. In the same vein, an account of computation arguably must also characterize miscomputation; that is, a malfunction where the operation of the system violates a norm related to the task it is performing (Piccinini, 2015; Tucker, 2018). Not all accounts do so. Yet, it is difficult to ignore miscomputation in the case of real-time computing, because the sort of predictability required to meet deadlines is baked into the definition of the system itself.

Another notable difference between real-time computing and standard discussions of implementation is that it is not obviously *medium-independent* (Haugeland, 1989). An influential, generic characterization of physical computation is that it is medium-independent because whether a physical system carries out some abstract computation depends solely on whether its states have appropriate degrees of freedom to map onto the vehicles of the computation

[9]Size may be such an example, though we are inclined to think that it is parasitic on the requirement that real-time systems meet their deadlines. If a computing device must meet certain size restrictions to interface with other systems, then this is a byproduct of the requirement of predictable, accurate operation.

[10]This claim differs in its severity. At one extreme is unlimited pancomputationalism, according to which all physical systems (with sufficient complexity) carry out a large number of computational operations (Putnam, 1988; Searle, 1992). According to limited pancomputationalism every physical system performs at least one computation (Chalmers, 2011; Scheutz, 2001).

and the rules that are defined over them (Piccinini and Scarantino, 2011; Piccinini, 2015). As long as the rules are respected, it does not matter what a physical system is made of. But note that this assumes a characterization of computing tasks where the correctness is only a matter of the result that is produced. In real-time computing, deadlines are defined relative to the other systems with which a computer interfaces. Thus computing tasks are in part a hardware question, as is abundantly clear in the case of the Atari 2600 port of Pac-Man. So in short, physical computation, when it comes to real-time computing, cannot be characterized as medium-independent, even if it is nonetheless multiply realizable.

Here an analogy might be helpful. As others have noticed, while the distinction between structure and function is intuitive (Fodor, 1968), and functional properties may not be defined explicitly *in terms* of a medium, they may still be defined *relative* to one (Kalke, 1969; Piccinini and Craver, 2011). To use Fodor's (1968) classic example, "valve lifter" may be a functional property, but any artefact that can fulfill that function in an internal combustion engine will share structural properties with camshafts. Similar considerations apply to computing tasks with deadlines and the embedded systems that carry out the tasks. A program for Pac-Man can be abstractly defined and so floats free of a particular medium. But it cannot be used to actually *play Pac-Man* unless it can be implemented in a way that ensures the required predictability for updating the playfield graphics inline with the timing requirements of the hardware.

Note that rejecting the medium-independence of some physical computations is not as strange as it may at first appear. For example, Maley (2021) argues that analog representation is inherently tied to a medium and so analog information-processing does not have a medium-independent representational level, while Curtis-Trudel (2021) proposes a view of implementation as resemblance that rejects the idea that computation by physical systems is medium-independent. Thus, if we are right that implementation, in the case of real-time computing, is not medium-independent, then this is in line with a recent trend towards rethinking the traditional divide between abstract and concrete computation and the corresponding idea that implementation is a form of mapping between the two.

5.2 Computational explanation in cognitive science

The final important implication concerns computational approaches to explanation in cognitive science. One objection leveled against such approaches is that they abstract away from the real-time occurrences of mental processes. Thus, computation is not the appropriate explanatory framework for cognitive science. Instead approaches grounded in dynamic systems modeling should be favored (Van Gelder, 1998; Van Gelder and Port, 1995). Replies have taken on different forms (Rescorla, 2020). One response is to maintain that timing is solely a feature of implementation details, which are separate from the aspects of mental processes that are the target of computational explanation (Weiskopf, 2004; cf. Pylyshyn, 1979, 1984). Another is to emphasize that timing matters a great deal to physical computation, but only insofar as it constrains the mapping between abstract computations and a physical system

(Clark, 1998; Piccinini, 2010). Still others have tried to add an overt dynamic element to the formal machinery of to computational explanations of mental processes (Eliasmith, 1996, 2003; Vera and Simon, 1993).

Despite their differences, each of these replies treats timing as either irrelevant, marginal, or supererogatory when it comes to computational explanation. Our discussion of real-time computing suggests a simpler response. For we have shown that it is simply mistaken to hold that computational explanations must inherently abstract away from real-time duration of events. On the one hand, we have shown—using an example of a paradigmatic digital computer—that some computational explanations can, do, and must take account of real-time. So at most the objection is premised on characterizations of computational explanation that need not be adopted. On the other hand, the premise that the real-time operations of mental processes are important explananda for cognitive science can be re-described as the claim that the brain is an embedded computing system with real-time deadlines. So rather than providing an objection against computational explanation in favor of dynamical explanation, highlighting the importance of timing to explaining mental processes can be recast as an argument in favor of real-time computational explanation.

We suspect that some proponents of computational—and even dynamical—explanation in cognitive science would be happy to accept such a conclusion. Others might balk: there is a strong tradition which characterizes computational and also dynamical explanation solely in abstract terms, and it can be hard to shake off those intuitions.[11] Yet if the brain is an embedded computational system, for which timing is crucial to how it represents and interacts with the world, then a wholly abstract approach will be inadequate. An account of the mind that leaves such details out will have as much utility as an explanation of the video game Pac-Man that simply specifies the rules of the game, but is silent about how the game actually works, or why certain differences obtain.

6 Conclusion

The foregoing discussion does not by any stretch exhaust the philosophical import of real-time computing. Each of the implications we have suggested can and should be considered and challenged. But we hope to have shown that, much like complexity theory (Aaronson, 2013), real-time computing is a branch of computer science that philosophers should take seriously. Expanding computational explanation to encompass real-time computing provides an account commensurate with the rich variety of embedded computational systems that surround us.

[11]For examples of such a view among proponents of dynamical approaches, see: Barack (2019); Meyer (2020); Ross (2015); Silberstein and Chemero (2013); Walmsley (2008).

References

Aaronson, S. (2013). Why philosophers should care about computational complexity. *Computability: Turing, Gödel, Church, and Beyond*, 261:327.

Barack, D. L. (2019). Mental machines. *Biology & Philosophy*, 34(6):1–33.

Buttazzo, G. C. (2011). *Hard real-time computing systems: predictable scheduling algorithms and applications*, volume 24. Springer Science & Business Media.

Chalmers, D. J. (2011). A computational foundation for the study of cognition. *Journal of Cognitive Science*, 12(4):325–359.

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

Chirimuuta, M. (2020). Explanation in computational neuroscience: Causal and non-causal. *The British Journal for the Philosophy of Science.*

Clark, A. (1998). Time and mind. *The Journal of Philosophy*, 95(7):354–376.

Coelho Mollo, D. (2018). Functional individuation, mechanistic implementation: The proper way of seeing the mechanistic view of concrete computation. *Synthese*, 195(8):3477–3497.

Cummins, R. (1989). *Meaning and mental representation.* MIT Press, Cambridge.

Curtis-Trudel, A. (2021). Implementation as resemblance. *Philosophy of Science*, 88(5):1021–1032.

Curtis-Trudel, A. (2022). The determinacy of computation. *Synthese*, 200(1):1–28.

Dewhurst, J. (2018). Computing mechanisms without proper functions. *Minds and Machines*, 28(3):569–588.

Eliasmith, C. (1996). The third contender: A critical examination of the dynamicist theory of cognition. *Philosophical Psychology*, 9(4):441–463.

Eliasmith, C. (2003). Moving beyond metaphors: Understanding the mind for what it is. *The Journal of philosophy*, 100(10):493–520.

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, Cambridge.

Fresco, N. (2021). Long-arm functional individuation of computation. *Synthese*, 199(5):13993–14016.

Gallistel, C. R. and King, A. P. (2011). *Memory and the computational brain: Why cognitive science will transform neuroscience.* John Wiley & Sons.

Harbecke, J. and Shagrir, O. (2019). The role of the environment in computational explanations. *European Journal for Philosophy of Science*, 9(3):1–19.

Haugeland, J. (1989). *Artificial intelligence: The very idea.* MIT press, Cambridge.

Hooker, S. (2021). The hardware lottery. *Communications of the ACM*, 64(12):58–65.

Kalke, W. (1969). What is wrong with Fodor and Putnam's functionalism. *Nous*, pages 83–93.

Kaplan, D. M. (2011). Explanation and description in computational neuroscience. *Synthese*, 183(3):339–373.

Klein, C. (2020). Polychronicity and the process view of computation. *Philosophy of Science*, 87(5):1140–1149.

Lee, E. A. (2008). Cyber physical systems: Design challenges. In *2008 11th IEEE international symposium on object and component-oriented real-time distributed computing (ISORC)*, pages 363–369. IEEE.

Lee, E. A. (2018). What is real time computing? A personal view. *IEEE Des. Test*, 35(2):64–72.

Lee, J. (2020). Mechanisms, wide functions, and content: Towards a computational pluralism. *The British Journal for the Philosophy of Science.*

Maley, C. J. (2021). The physicality of representation. *Synthese*, pages 1–26.

Marr, D. (1982). *Vision.* Freeman and Company, San Francisco.

Meyer, R. (2020). The non-mechanistic option: Defending dynamical explanations. *The British Journal for the Philosophy of Science.*

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

Montfort, N. and Bogost, I. (2009). *Racing the beam: The Atari video computer system.* MIT Press, Cambridge.

Newell, A. (1990). *Unified theories of cognition.* Harvard University Press.

Papayannopoulos, P., Fresco, N., and Shagrir, O. (2022). On two different kinds of computational indeterminacy. *The Monist*, 105(2):229–246.

Piccinini, G. (2007). Computing mechanisms. *Philosophy of Science*, 74(4):501–526.

Piccinini, G. (2010). The resilience of computationalism. *Philosophy of Science*, 77(5):852–861.

Piccinini, G. (2015). *Physical computation: A mechanistic account.* OUP Oxford.

Piccinini, G. and Craver, C. (2011). Integrating psychology and neuroscience: Functional analyses as mechanism sketches. *Synthese*, 183(3):283–311.

Piccinini, G. and Scarantino, A. (2011). Information processing, computation, and cognition. *Journal of biological physics*, 37(1):1–38.

Putnam, H. (1988). *Representation and reality*. MIT press.

Pylyshyn, Z. W. (1979). Do mental events have durations? *Behavioral and Brain Sciences*, 2(2):277–278.

Pylyshyn, Z. W. (1984). *Computation and cognition*. MIT press Cambridge, MA.

Rescorla, M. (2020). The Computational Theory of Mind. In Zalta, E. N., editor, *The Stanford Encyclopedia of Philosophy*. Metaphysics Research Lab, Stanford University, Fall 2020 edition.

Ritchie, J. B. (2019). The content of marr's information-processing framework. *Philosophical Psychology*, 32(7):1078–1099.

Ritchie, J. B. and Piccinini, G. (2018). Computational implementation. In *The Routledge handbook of the computational mind*, pages 192–204. Routledge.

Ross, L. N. (2015). Dynamical models and explanation in neuroscience. *Philosophy of Science*, 82(1):32–54.

Scheutz, M. (2001). Computational versus causal complexity. *Minds and Machines*, 11(4):543–566.

Searle, J. R. (1992). *The rediscovery of the mind*. MIT press.

Shagrir, O. (2001). Content, computation and externalism. *Mind*, 110(438):369–400.

Shagrir, O. (2010). Marr on computational-level theories. *Philosophy of Science*, 77(4):477–500.

Shagrir, O. (2020). In defense of the semantic view of computation. *Synthese*, 197(9):4083–4108.

Shin, K. G. and Ramanathan, P. (1994). Real-time computing: A new discipline of computer science and engineering. *Proceedings of the IEEE*, 82(1):6–24.

Silberstein, M. and Chemero, A. (2013). Constraints on localization and decomposition as explanatory strategies in the biological sciences. *Philosophy of Science*, 80(5):958–970.

Simon, H. A. (1992). What is an "explanation" of behavior? *Psychological science*, 3(3):150–161.

Stankovic, J. A. (1988). Misconceptions about real-time computing: A serious problem for next-generation systems. *Computer*, 21(10):10–19.

Tucker, C. (2018). How to explain miscomputation. *Philosophers Imprint*, 18.

Van Gelder, T. (1998). The dynamical hypothesis in cognitive science. *Behavioral and brain sciences*, 21(5):615–628.

Van Gelder, T. and Port, R. F. (1995). Dynamical approach to cognition. *Mind as motion: Explorations in the dynamics of cognition*, page 1.

Van Rooij, I. (2008). The tractable cognition thesis. *Cognitive science*, 32(6):939–984.

Vera, A. H. and Simon, H. A. (1993). Situated action: A symbolic interpretation. *Cognitive science*, 17(1):7–48.

Walmsley, J. (2008). Explanation in dynamical cognitive science. *Minds and Machines*, 18(3):331–348.

Weiskopf, D. A. (2004). The place of time in cognition. *The British journal for the philosophy of science*, 55(1):87–105.

Weiskopf, D. A. (2011). Models and mechanisms in psychological explanation. *Synthese*, 183(3):313–338.