Pancomputationalism and the Computational Description of Physical Systems[1]

Neal G. Anderson
University of Massachusetts Amherst

Gualtiero Piccinini
University of Missouri – St. Louis

Abstract – According to pancomputationalism, all physical systems – atoms, rocks, hurricanes, and toasters – perform computations. Pancomputationalism seems to be increasingly popular among some philosophers and physicists. In this paper, we interpret pancomputationalism in terms of computational descriptions of varying strength—computational interpretations of physical microstates and dynamics that vary in their restrictiveness. We distinguish several types of pancomputationalism and identify essential features of the computational descriptions required to support them. By tying various pancomputationalist theses directly to notions of what counts as computation in a physical system, we clarify the meaning, strength, and plausibility of pancomputationalist claims. We show that the force of these claims is diminished when weaknesses in their supporting computational descriptions are laid bare. Specifically, once computation is meaningfully distinguished from ordinary dynamics, the most sensational pancomputationalist claims are unwarranted, whereas the more modest claims offer little more than recognition of causal similarities between physical processes and the most primitive computing processes.

1. Introduction

An ordinary rock implements every finite-state automaton. A pail of water is, for fleeting moments, computationally equivalent to a conscious human brain. The universe is a computer. Everything computes.

Such claims, which are increasingly common in both the philosophical and scientific literature, are closely related to the broad thesis that all physical systems perform computations: pancomputationalism (PC). Any pancomputationalist claim rests on some notion of what it means for a physical system to implement a computation, and may be taken as patently absurd, empirically substantive, or trivially self-evident depending on what counts physically as a computation.

[1] Version 2.8.17. The authors thank Mike Cuffaro, Ilke Ercan, Samuel Fletcher, Nir Fresco, Weibo Gong, John Norton, Jack Mallah, Marcin Milkowski, Steve Selesnick, Oron Shagrir, and an anonymous referee for comments and suggestions on an earlier version of this manuscript, some of which have been only partially addressed at this stage. We also thank the participants in the Nature as Computation Workshop (Tempe, 2015)—especially Paul Davies, Gregory Chaitin, and James Crutchfield—for helpful discussion after a presentation of this work.

In this work we adopt our earlier classification of pancomputationalism into three generic types (Piccinini 2015) – unlimited PC, limited PC, and ontic PC. We then identify three general classes of computational description—three explicit specifications of what counts as computation in physical systems that vary in their restrictiveness—and characterize the various types of PC in terms of the nature and strength of the computational descriptions required to support them. This allows pancomputationalist claims to be evaluated in terms of what they inherently say about physical computation itself.

We set the stage by considering computational states, computational processes, and usability, and associating the strength of a computational description with its commitments regarding these fundamental notions (Section 2). We then consider unlimited PC (Sec 3(a)). We identify several variants of unlimited PC and show that they are supported only by the weakest form of computational description. Next, we take up limited PC (Sec. 3(b)). We show that trivial forms of limited PC survive when the restrictions on computational description that support unlimited PC are expanded moderately – but only moderately – and argue that less trivial forms of limited PC are rendered untenable by these additional restrictions. Here we address unique issues that arise when the would-be computing system is the universe as a whole. Finally, we briefly consider ontic PC (Sec. 3(c)), which largely sidesteps the requirements for computational description and is evaluated on other grounds elsewhere (Piccinini and Anderson 2017).

Five aspects of our study should be noted:

First, we avoid the common formulation of PC to the effect that everything is a *computer*. The reason is that in both computer science and common parlance the term "computer" usually refers to computing systems with special features such as programmability and computational universality (up to their memory limitations) in Turing's (1936-7) sense. Many computing systems are not programmable or computationally universal, so many computing systems are not *computers* in this sense. Instead, we formulate pancomputationalism as the claim that everything *performs computations* or, equivalently, everything *is a computing system*.

Second, in considering pancomputationalism, we accept that there is a sense in which a physical system may perform computations even though it has no semantic properties and does not have the *function* to compute. According to semantic accounts of computation, there is no computation without representation; according to mechanistic accounts, computing systems are physical systems whose *function* is to perform computations (Piccinini 2015). Given these accounts, anything that is has no semantic properties or no functions, respectively, is ruled out of the class of computing systems. So pancomputationalism is false according to these accounts. This may be acceptable in other contexts but is too strong for present purposes. Since we wish to entertain pancomputationalism as a serious possibility and give it due process, we admit accounts of physical computation that do not rule out pancomputationalism almost from the start.

Third, we will focus on *literal* pancomputationalism – the thesis that all physical systems *do* perform computations. Except where otherwise noted, we set aside *metaphorical* pancomputationalist theses to

the effect that all physical systems *can be regarded* or *interpreted* as computational. Metaphorical PC may be useful and may open perspectives that yield valuable insights in both physics and computation, but it says something fundamentally different and much less sensational than literal PC.

Fourth, we focus primarily on pancomputationalism related to deterministic classical (as opposed to quantum) digital computation in systems with at least quasi-classical physical descriptions, as this is the most common notion employed by pancomputationalists. Unless otherwise noted, for present purposes we take a digital computing system to be a finite-state automaton, described abstractly by countable sets of computational states C, inputs I, transition rules (C,I)→C' that govern the automaton's behavior, and outputs O(C) (or O(C,I)). Quantum computation will also be discussed in connection with ontic PC. Much of our analysis would also apply, mutatis mutandis, to a hypothetical version of pancomputationalism formulated in terms of classical analog computation (Pour-El 1974, Rubel 1989, Mills 2008) and to probabilistic classical digital computation.

Fifth and finally, we emphasize that by "computational description of a physical system" we mean ascription of would-be computational states and processes to physical microstates and their dynamics, taking the microphysical to be primary and given. This is to be distinguished from what might be called a physical description of a computational system, i.e., a description of a specified system or artifact—deemed "computational" at the outset—that provides a physical account of the system's presumed computational capabilities. Descriptions of the latter sort, which can take highly sophisticated forms for specific types of computing devices and are stock-in-trade for the engineers that create them, are ill suited for analysis of pancomputationalism. Evaluation of claims about the inherent computational capacities of arbitrary physical systems requires the former sort of description—a description that starts with physical microstates and their dynamics and delineates possible options for their connection to computational states and processes. Such are the descriptions we employ in this work, as we now discuss in detail.

2. Computational Description

What counts as a computational description of a physical system? More precisely, what sort of relation between physical microstates and their dynamics, on one hand, and computational states, computational processes, and interaction with users, on the other hand, renders physical systems computational? There are many possible answers to this question, some compatible with particular varieties of pancomputationalism and some incompatible with pancomputationalism in any form. In this section we identify classes of possible computational descriptions, organized by the restrictiveness of the commitments they require concerning the correspondence between the microphysical and the computational. These classes of computational description will be used in Section 3 to analyze varieties of pancomputationalism, highlighting the restrictiveness of the commitments they require.

2(a). Elements of Computational Description

We begin by discussing elements of computational description and identifying possible commitments that one might make regarding their correspondence with microphysical description.

Computational States

In principle and in practice, there is a natural correspondence between physical microstates and computational states. The simplest possible correspondence would be a one-to-one mapping between the physical microstates of a system and its computational states. But ordinary computational descriptions posit countably many computational states, while ordinary (continuous) microphysical descriptions posit uncountably many states. Therefore, ordinarily there aren't enough computational states for a one-to-one mapping with physical microstates. Alternatively, one can select a countable subset of the physical microstates and map the computational states onto them. More realistically, computational states are taken to correspond to disjunctions of physical microstates, regions of the physical system's state space (each containing an uncountably infinite number of microstates), or – most generally – disjunctions of these state-space regions. With this, a minimal commitment regarding physical-computational state correspondence is the following:

> **S**: Every computational state corresponds to a distinct physical microstate of a system, a disjunction of physical microstates, a region of the system's state space, or a disjunction of non-overlapping state-space regions.

Note that, in whichever manner **S** is satisfied, no physical microstate can be shared by more than one computational state if all computational states are to be perfectly distinguishable from one another, and we will take this to be implicit in **S**. Note also that many physical microstates may go without any computational state assignment at all. This is natural and common in ordinary computing systems; the microstates accessed by a system during transitions between computational states, for example, may not themselves be associated with any computational state. The physical state of a system need only correspond to a computational state during time intervals deemed to be "computationally relevant times."

We emphasize that, by condition **S** alone, a physical microstate may encode much more than the computational state with which it is associated. Given knowledge of a system's dynamics, or, alternatively, a complete record of the sequence of microstates visited by a system with unknown dynamics, specification of the system's microstate may be used to identify predecessor microstates. This has nontrivial implications for computational descriptions that require nothing more than **S** when associating physical states with computational states. To illustrate this, suppose that at time t the physical microstates P(t) and Q(t) lie in the region of the state space corresponding to a given computational state C. Suppose also that P(t) and Q(t) are known to have evolved from – or to

necessarily evolve from – microstates P(t-1) and Q(t-1), respectively, and that P(t-1) and Q(t-1) lie in regions of the system's state space that correspond to *different* computational states C'_A and C'_B. Specification of the system's physical microstate at time *t* as P(t) implies not only that the system is in computational state C at time t, but also that it was in computational state C'_A (and not C'_B) at time *t*-1. In a computational description, however, specification of a system's computational state as C at time *t* would imply that the system *could have been* in either C'_A or C'_B at *t-1* but would not identify the predecessor as C_A'.

This implicit disjoining of physical microstates *and their predecessor states* into physical realizations of static computational states goes beyond a simple correspondence between computational states and physical states, and is not prevented by **S** alone. A physical microstate taken (via **S**) to correspond to a given computational state may,, under some dynamics, actually correspond to a computational state together with its predecessor physical microstate. In such cases, a physical microstate can encode both a computational state *and* its computational predecessor state, even when there is more than one possible computational predecessor state.

The admissibility of such *representationally unfaithful* mappings[2] – mappings that allow excess information about the history of a computational state to be encoded in the fine-grained structure of its physical representation – has nontrivial consequences for implementation. For example, in the absence of a requirement for faithful representation, there is no objective basis for distinguishing an evolution of a physical system that maps each of M computational states onto itself (i.e., it does nothing computationally) from an evolution that maps M computational states into N computational states with N<M (i.e., implements a logically irreversible transformation). The dissipative cost associated with physical implementation of logically irreversible transformations, reflected in Landauer's Principle (Landauer 1961, Bennett 2003), can in fact be understood as the physical cost of generating representationally faithful output states in such implementations (see Anderson 2010).

Thus, in constructing a computational description of a physical system, one may reasonably go beyond a simple correspondence between computational states and collections of a system's microstates (i.e., **S**) and also insist that nothing about the computational history – about the sequence of predecessor computational states (including the initial state) – can be inferred from any physical microstate that cannot also be inferred from the computational state to which this physical microstate belongs. Supplementing **S** with such a requirement – a requirement that computational states have *faithful* physical realizers – yields a much stronger commitment concerning physical-computational state correspondence:

> **S'**: Every computational state corresponds to a distinct physical microstate of a system *or to a statistical state* defined on a disjunction of physical microstates, on a region of the system's state space, or on a disjunction of non-overlapping state-space regions.

[2] The notion of faithful representation was discussed by (Ladyman 2007) in the context of classical thermodynamic systems that implement logical transformations (classical L-machines). This notion was generalized and formalized, and a quantitative faithfulness measure introduced, for generally noisy, generally quantum-mechanical L-machines in (Anderson 2010).

Critically, by "statistical state" we mean an assignment of a probability p(s,C)—at computationally relevant times— to each microstate s associated with the computational state C, with no conditioning on predecessor computational states or the microstates that belong to them.[3] This is to say that all transitions into computational state C ultimately "land in" physical microstate s with probability p(s,C) regardless of their source microstate, and that all transitions out of computational state C "depart from" microstate s with probability p(s,C) regardless of their destination microstate. Stated more formally, if computational state C evolved from predecessor computational state C', then (i) every microstate s in C evolved from microstate s' in C' with probability p(s',C') and (ii) every microstate s' in C' evolves to microstate s in C with probability p(s,C). Similarly, if computational state C evolves to successor computational state C'', then (i) every microstate s'' in C'' will evolve from every microstate s in C with probability p(s'', C'') and (ii) every microstate s in C will evolve to microstate s'' in C'' with probability p(s,C).[4]

If the dynamics of a physical system are such that this condition cannot be met for a given mapping onto a computational description, at least at or during time intervals for which the computational states are defined (i.e., at "computationally relevant" times), then commitment to **S'** would require rejection of such a mapping. If the dynamics of a physical system are such that this condition cannot be met for *any* nontrivial mapping – i.e., any mapping that disjoins multiple physical microstates into computational states – then commitment to **S'** rules out any such mapping.

A few remarks are in order regarding **S'**. First, it may appear that **S'**, which is supposed to be about computational states, is about both computational states *and* computational processes since it implicitly requires that the dynamics randomize microstates "within" the state subspaces belonging to the various computational states. On the contrary, it is this condition that selects only microphysical-to-computational state mappings for which physical microstates encode *nothing more* than computational states. It precludes association of a physical microstate with a computational state if the microstate encodes information about the system's dynamical history that is more finely grained than that available from an abstract computational description (i.e. description of the system's dynamics at the level of the computational states alone).

[3] In a quantum-physical description of these computational states, the role of p(s,C) is played by a density operator or density matrix—specifically a reduced density operator or density matrix if the system is not isolated from its environment.

[4] Here the introduction of probabilistic state descriptions recognizes that physical systems generally interact uncontrollably with their surrounding environments, and that such system-environment interactions result in system dynamics that are partially nondeterminisitic. This applies to most systems considered by pancomputationalists (e.g., rocks and pails of water), with the exception of the universe as a whole (discussed in Sec. 3(b)). Randomness in dynamical evolution necessitates statistical description of states that evolve from a specified initial state or from initial states within a specified state subspace. Microstate-level physical descriptions of computational states for systems that interact with environments are thus necessarily statistical, even when transitions between the computational states are deterministic (see text). Statistical state description does no violence to systems that are isolated from their environments and thus evolve deterministically, since deterministic evolution is a special case of stochastic evolution where the probability is unity for single dynamical microstate trajectory and vanishes for all other trajectories.

Second, note that the probabilistic nature of **S'** does not in any way preclude its application to deterministic computation. **S'** generalizes **S** to allow for microscopically stochastic processes that randomize (i) the physical microstates *within* the region of a system's state space that corresponds to a given computational state and/or (ii) the physical microstate-to-microstate transitions that enable computational state transitions by bridging the state-space regions corresponding to computational states and their deterministically specified successors. Such randomizing processes, which can occur only in systems interacting with an environment, are required for computational states corresponding to more than one physical microstate to "forget" enough about their detailed dynamical history to have faithful microphysical realizers. These processes are taken for granted in standard notions of computational state realization for conventional computing devices. In the registers used in deterministic digital computers, for example, where the register state is encoded in the coarse-grained spatial distribution of charge density, nothing in the microstate of the system of charged particles—the detailed roster of all electron positions and velocities in the register system—is presumed to encode anything more than the register state at times that the register state is considered "valid" (i.e., at computationally relevant times).

Third, we note that, in typical computing systems, the microphysical state space corresponding to any given computational state is vast, and it is entirely reasonable to expect that, throughout the entire life of the system, (i) no microstate belonging to any computational state will be visited more than once, and (ii) an exceedingly small fraction of the microstates corresponding to this computational state will ever be visited. Thus, for macroscopic physical computing systems, it is not plausible that satisfaction of **S'** could be empirically confirmed from a record of its dynamical history even if the microstate's trajectory through the state space could be precisely tracked. This precludes the possibility, even in principle, of building "sufficient statistics" for empirical confirmation of **S'**; acceptance of **S'** for a given system thus relies on acceptance that physical laws are such that it will be satisfied – at least to a sufficiently good approximation – under operating conditions of interest. Nothing that follows relies on direct empirical confirmation of **S'**, despite the necessity of **S'** for establishing a one-to-one correspondence between computational states and (generally statistical) physical states. It is equally important, particularly for the discussion of unlimited pancomputationalism in Sec. 3(a), that nothing in **S'** *precludes* any given microstate from being revisited when the computational state to which it belongs is revisited in the course of a computation. By **S'**, when any given computational state is visited more than once, any given microstate belonging to that computational state may or may not be revisited.

Fourth, note that **S'** could be satisfied by systems with *completely* random dynamics, in which every physical microstate evolves from every other microstate with equal probability, if we allow for computational descriptions with probabilistic state transitions. Even though our concern is deterministic machines, this observation is a red flag that motivates the additional restrictions on computational descriptions discussed below.

Computational Processes

If physical microstates are associated with computational states, then transitions between computational states naturally require time evolution of physical microstates. In a computational

description of a physical system that involves a finite number of computational states, each associated with a region of the system's physical state space, transitions between different computational states are specifically associated with families of physical microstate transformations that evolve microstates belonging to one computational state into microstates belonging to another. A minimal commitment regarding the correspondence between physical and computational processes is this:

> **P**: Every computational state transition corresponds to one or more time evolutions of a system's physical microstate along physically possible state-space trajectories, possibly restricted to specific time intervals.

Here, by a "physically possible state-space trajectory", we mean a state-space trajectory that is lawfully consistent with the system's dynamics and the external influences (applied and/or environmental) that act on the system. The temporal restriction allows computational processes to be defined only over time intervals that bridge the "computationally relevant" time intervals over which computational states are defined.

Note that, by **P** alone, the evolution of any physical system that takes its physical microstates along a trajectory passing through microstates or state space regions that have been assigned to computational states would qualify as a computational process, since any such trajectory necessarily corresponds to *some* sequence of computational states. Consider a leaf falling from a tree. Different positions of the leaf may be mapped onto different computational states and, by **P** alone, the leaf would generate a succession of computational state transitions simply by floating to the ground.

An obvious and reasonable objection is that **P** cannot capture important features of computational processes, such as the conditional dependence of computational state transitions and outputs on computational inputs and previous computational states. Philosophers have gone to surprising lengths in trying to accommodate these features, or at least the observational consequences of these features, within computational descriptions that require only **P** in relating the dynamics of physical systems to computational processes (cf. Putnam 1988, Chalmers 1996; more on this below). The resulting computational descriptions are not, however, counterfactually robust in the right way, as they fail to capture the kind of causal structure that characterizes bona fide computational processes. In our falling-leaf example, whether the leaf ends up at a certain location depends not only on where it was before (its previous "computational" state), but also on how it is oriented and how its shape interacts with the random wind currents that blow it about. There is no way to determine *a priori* where the leaf would go from a certain position (its "computational" state transition) solely on the basis of its position, velocity, and orientation (its "computational" state).

Thus, in constructing a computational description of a physical system, one may reasonably require that one go beyond a simple correspondence between computational state transitions and physical microstate transitions (i.e., **P**) and also require that computational state transitions correspond to physical state transitions that are causal and counterfactual-supporting and, if the system receives inputs, are affected by the inputs in the right way. By "inputs" we mean external influences on the system that conditionally alter its internal dynamics to produce computational state transitions, and we

will similarly define "outputs" as influences the system exerts on the outside world that depend exclusively on the system's computational state or on transitions the system undergoes between computational states. These definitions of inputs and outputs, while very general, are sufficient for present purposes; they will be expanded upon later in this section.

Now, supplementing **P** with such requirements on state transitions yields a much stronger commitment concerning physical-computational process correspondence:

> **P'**: Every computational state transition corresponds to one or more time evolutions of a system's physical microstate along physically possible state-space trajectories, possibly restricted to specific time intervals, such that during the relevant time intervals each physical microstate that corresponds to a given computational state evolves from (into) a microstate that corresponds to its predecessor (successor) computational state as specified by a set {(C,I)→C'} of deterministic state transition rules.

If the dynamics of a physical system are such that – for a given mapping from physical microstates to computational states – there exist no nontrivial sets {(C,I)→C'} of state transition rules for which this condition is satisfied, then commitment to **P'** requires rejection of such a mapping. If the dynamics of a physical system are such that this condition cannot be met for *any* such mapping for any nontrivial set of state transition rules, then commitment to **P'** is untenable and possible computational descriptions of the system are limited to those sufficiently weak to require nothing more restrictive than **P**.

A few remarks concerning **P'**. First, we saw that **S'** adds to **S** that the microstates contain *no more* information about the system's computational history – i.e., predecessor computational states and inputs specified by {(C,I)→C'} – than is encoded in the computational states themselves. **P'** adds the further restriction that the microstates contain *at least as much* information about the system's computational history as is encoded in the computational states. We say that computational descriptions satisfying this condition exhibit *perfect computational fidelity*.[5] State transition rules in such descriptions are necessarily deterministic at the level of computational states but need not be at the level of physical microstates.

Second, note that **S'** and **P'** jointly ensure that the computational description and the microphysical description contain exactly the same information about the computational process {(C,I)→C'}. This is equivalent to saying that the computational description captures an objective aspect of the system's causal structure—the physical microstates that correspond to each computational state all share causal powers whose effects are captured by the computational description. We call this condition *robustness*.[6]

[5] Like the representational faithfulness, a quantitative measure for the computational fidelity has been defined for physical implementation of classical logical transformations in generally noisy, generally quantum mechanical systems (quantum L-machines) in (Anderson 2010); both measures are unity for systems with computational descriptions that satisfy **S'** and **P'**.

[6] Robustness comes at a dissipative physical cost, which is lower bounded for physical systems that implement logical transformations—robust and otherwise—in (Anderson 2010) and for robust physical FSA in (Ganesh 2013).

Robust descriptions support all and only the counterfactuals captured by the computational description, whereas non-robust descriptions fail to support such counterfactuals. For example, consider a system in physical microstate s, corresponding to computational state C, which evolves from predecessor state C' into successor state C''. **S'** (faithfulness) guarantees that all computationally relevant microstates of the system encode no more information about the system's dynamical history than is contained in the computational state transition rules; **P'** (fidelity) guarantees that they encode all such information. Therefore, if the system had been in any other arbitrary state s' corresponding to computational state C''' (possibly identical to C), a robust description guarantees that it would have evolved from the computational state that precedes C''' and would have evolved into the computational state that succeeds C'''. If **P'** (fidelity) is not satisfied, the right counterfactuals may not be satisfied; if the system had been in s', it may have evolved from (evolve into) any arbitrary computational state. If **S'** (faithfulness) is not satisfied, the wrong counterfactuals may be satisfied; if the system had been in s' where s' also corresponds to C (as s does), it may be guaranteed to have evolved from a specific computational state C'''' distinct from C', even though nothing in the computational state transitions distinguishes between s and s' in this way. This shows that **S'** and **P'** jointly ensure that a computational description captures an objective aspect of the causal structure of a physical system in a counterfactual-supporting way.

Usability

In attempting to single out physical systems that are truly computational, one may regard criteria tying computational states and processes to their physical counterparts as insufficient and also require that a system be *usable* for computation by an agent (cf. Piccinini 2015, Hughes 2009, Houkes & Vermaas 2010, Anderson 2016a). Specifically, one may require a use plan that would enable an agent to use the system to evaluate a function (or functions) for arguments of the agent's choosing. Note that such a requirement does not preclude consideration of natural systems as computational systems, since use plans may exist for harnessing computational capacities of natural systems appropriated for computational purposes as well as human-made computing machines.[7]

A use plan, among other things, must specify the actions an agent would have to take to enter computational inputs for evaluation of a specified function, commence the computational process (if it does not occur spontaneously), and read the resulting computational outputs. This implies specification of physical degrees of freedom that are associated with computational inputs and outputs, as well as user accessibility to these degrees of freedom so the required input states can be prepared and appropriate measurements can be selected to read output states. It also implies *ex ante* assignment of all computational inputs and outputs to specified physical system (or subsystem) states (or state-space regions) and/or user-system interactions required to prepare and ascertain these states.

With these considerations in mind, we may formulate a commitment on usability:

[7] Nevertheless, the existence of agents using a natural system according to use plan turns the system into something that has functions in the sense in which biological systems and ordinary artifacts have functions. Thus, a natural system with a use plan satisfied the mechanistic account of computation (Piccinini 2015).

U: A usable computing system is a physical system that (a) supports computational processes on computational states, and (b) for which there exist use plans that would enable an agent to use the system to evaluate a function or functions for arguments of their choosing.

Note that acceptance of **S** (or **S'**) and **P** (or **P'**) is implicit in **U**. Elsewhere (Anderson 2016b), we have called systems that satisfy (a) *protocomputing systems* and systems that satisfy both (a) and (b) *computing artifacts*.

2(b). Classes of Computational Description

We now identify and discuss three broad classes of computational description, defined and ordered according to the strictness of their commitments. These classes will guide our evaluation of pancomputationalist theses in the following section.

Weak: A computational description of a physical system is *weak* if it satisfies only **S** and **P, S'** and **P,** or **S** and **P'**.

Robust: A computational description of a physical system is *robust* if it satisfies **S'** and **P'** but not **U**.

Strong: A computational description of a physical system is *strong* if it satisfies **S'**, **P'**, and **U**.

The commitments **S** and **P** seem to be required for *any* computational description of a physical system, so computational descriptions that satisfy **S** and **P** (and nothing more) are absolutely minimal. The account of computation that requires these and only these minimal commitments is sometimes called the *simple mapping* account (Godfrey-Smith 2009). Slightly stronger computational descriptions that satisfy **S'** and **P** or **S** and **P'**, but not **S'** *and* **P'**, are also weak, simply because they fail either to require faithful physical representations of computational states or to provide a nontrivial specification of computational state transitions in terms of microphysical state transitions.

Robust computational descriptions are stronger, in that they include additional restrictions on the physical representation of computational states and/or the nature of the system's dynamics, but they do not include any usability criteria. As noted above, the robustness condition (**S'** *and* **P'**) that we require of robust descriptions harmonizes the computational and the microphysical, ensuring that specification of a system's physical microstate and state-space trajectory says no more and no less about the system's computational state and state transitions than does specification of the computational state and state transition to which that microstate and state-space trajectory belong.

Our robustness condition belongs to a family of conditions suggested by various authors to make computational description nontrivial. Specifically, counterfactual accounts of computation require that computational state transitions support counterfactuals (Block 1978, Maudlin 1989, Copeland 1996, Rescorla 2014); causal accounts require that computational state transitions mirror the causal structure of the physical system (Chrisley 1995, Chalmers 1995, 1996, 2011, Scheutz 1999, 2001); dispositional accounts require that computational state transitions follow from the system's dispositions (Klein 2008); a final account requires that the physical microstates correspond to a computational state must be similar to each other in a causally relevant way (Godfrey-Smith 2009). Our robustness condition improves upon each of these individual proposals by being more comprehensive—capturing the spirit of all of them—and by being more precise.[8]

Finally, we take strong computational descriptions to be those that require **P'** – the strongest commitment regarding computational processes – *and* the usability criterion **U**, which together with **S'** seem to constitute the minimal commitments required for computing artifacts such as computing machines. (Familiar electronic digital computing devices unambiguously admit strong computational descriptions.) We ignore computational descriptions that require **U** but only **P** or **S** (or both) on the following grounds: processes whose computational structure does not correspond to their physical causal structure – i.e., that do not satisfy **P'** – simply could not be used by agents to evaluate pre-specified functions for arbitrary inputs. Processes that satisfy **P'** but violate **S'** – so they do not produce faithful physical realizers of computational states – are usable in a strict sense but are not very useful as they leave computational work undone; the agent must do more to complete evaluation of the desired function. These exclusions will be substantiated in the section 3(a), where we give examples of systems that satisfy only **P'** but not **S'** (such as a falling apple) or vice versa (such as a random system). These systems clearly cannot be used to evaluate functions for arguments of one's choosing.

3. Varieties of Pancomputationalism

Pancomputationalist theses vary with respect to *how many* computations—all, many, a few, or just one—they attribute to each system. We now analyze unlimited and limited PC and the classes of computational descriptions that are required to support them. After that we discuss ontic PC, which is a kind of limited PC.

3(a). Unlimited PC

The strong version of pancomputationalism is unlimited PC (Putnam 1988, Searle 1992):

[8] Using the representational faithfulness and computational fidelity measures together, it is possible to quantify the degree to which the robustness condition is satisfied in concrete scenarios.

Unlimited PC: Every physical system performs every computation —or at least, every sufficiently complex system implements a large number of non-equivalent computations.

We now consider various forms of unlimited PC in detail.

The first argument for something like unlimited PC is known as Hinckfuss's pail, after its proponent Ian Hinckfuss. According to William Lycan's first published account of Hinckfuss's pail, a pail of water contains a large number of microscopic processes, which are complex enough to realize computations equivalent to those of a human mind—and by implication, any computation—at least for a brief time period (Lycan 1981, 39).

Other authors articulate similar arguments in greater detail. John Searle argues that whether a physical system implements a computation depends on how it is interpreted by an observer. Therefore, for any sufficiently complex system—such as a piece of wall with all its interacting particles—and any computation, the system can rightfully be regarded as performing that computation at least for a brief time period. Since Searle's conclusion is based on the premise that physical computation depends on the free interpretation of physical systems, it may be called *interpretive* unlimited PC.

Hilary Putnam gives a more rigorous argument for unlimited PC. With respect to finite automata without inputs and outputs, he argues that every ordinary open system implements every finite automaton (Putnam 1988, 122-3). With respect to finite automata with inputs and outputs, he argues that any physical system whose inputs and outputs are isomorphic to those of the finite automaton implements the finite automaton. Putnam's conclusion about automata with inputs and outputs is far weaker than his conclusion about automata without inputs and outputs, since, for nontrivial abstract automata, physical systems with inputs and outputs that can be regarded as being "isomorphic" to those of the abstract automata will be of an exceedingly special nature and thus not particularly relevant to consideration unlimited PC (which is supposed to apply to all physical systems). We thus focus on his arguments that are germane to FSA without inputs and outputs, except in a relatively weak sense to be discussed below.

Unlike Searle, Putnam does not appeal explicitly to the notion of interpretation in defending unlimited PC. But his argument relies on slicing and aggregating a system's dynamics in arbitrary ways. Each arbitrary slicing and aggregating is one of indefinitely many possible computational interpretations of the system. In this sense, Putnam's view is an example of interpretive unlimited PC. Indeed, some form of free computational interpretation of a physical system appears to be behind every defense of unlimited PC to be found in the literature, and it is unclear how else unlimited PC could be motivated. By the same token, the rejection of unlimited PC hinges on devising privileged, objective computational interpretations of a physical system.

If unlimited PC is correct, then the claim that a physical system performs a computation becomes almost trivially true and vacuous; it fails to distinguish that system from anything else (or perhaps from anything else with the same inputs and outputs). Because of this, unlimited PC is incongruous with computer science and engineering, where objective differences in the computational capacities of different

systems are critical and great effort is expended to realize physical systems with desired computational capacities. Where exactly, in this respect, does unlimited PC fall short?

Unlimited PC is Supported Only Under Weak Computational Descriptions

We will now argue that unlimited PC is supported only under weak computational descriptions because it violates **S'**, **P'**, or both. We show explicitly that this is the case for five different pancomputationalist claims.

Since we are going to discuss inputless and outputless FSA, however, we should first discuss the sense in which they can be said to evaluate a function defined over inputs. There are two ways to think about inputs and how FSAs can compute over them. The two ways are not completely distinct, but we'll talk about them separately for clarity. First, a string of inputs can be fed into an FSA as it is evolving, with (at least some) state transitions conditioned on some "incoming" input values until the desired computation has been completed. (This is expressed in the branching of FSA state diagrams.) Second, the same computation could be done over the same input string by a different FSA—an "inputless" FSA—by mapping all possible input strings into initial FSA states and letting the FSA evolve (without further input influence) until the computation has been completed. An example of the latter is a microprocessor that performs a computation on input data initially stored in its internal registers and then halts; no external inputs influence evolution of the processor state during the computation.

This second possibility opens the door to claims that FSA that are inputless (in the first sense) can perform computations over inputs or strings of inputs (in the second sense), with the crucial proviso that various initial states can be associated with various inputs. For strong computational descriptions, however, users must be able to prepare these initial states, which, alas, amounts to an external influence affecting at least one state transition—the "zeroeth" state transition—and becomes a special case of the first possibility. (In the above example, a user would have to be able to load data into the processor register states.) For weak and robust descriptions, instead, users need not be able to prepare these initial states – the initial states are assumed to be present regardless of any user's intervention.

Similarly, one can claim that outputless FSA perform computations by taking the final states to be encodings of the output. For example, a pancomputationalist could claim that the output of a computation performed by a falling leaf is instantiated in its final resting position on the ground, without being required to specify a unique logical mapping between the leaf position and some abstract symbol from an output set.

With this, we proceed to analyze several constructions and examples that motivate unlimited pancomputationalism.

Putnam's Construction

We begin with Putnam's construction, which aims to show that any sufficiently complex open system, such as a rock, implements any inputless and outputless finite-state automaton. His assumptions are minimal: First, electromagnetic and gravitational fields are assumed to be continuous. Second, physical systems are assumed to be in different maximal states—different physical microstates—at different times, with no microstate ever visited more than once as a system evolves along any dynamically supported trajectory. This second condition, which Putnam calls the Principle of Non-Cyclical Behavior, is assumed to hold for any system that cannot be completely shielded from the fields associated with external systems that have an arrow of time, and it effectively builds a "time stamp" into physical microstates.

Putnam considers a finite automaton that goes through the sequence of states ABABABA, and an arbitrary physical system S over the arbitrarily chosen time interval from 12:00 to 12:07 on an arbitrary day. Putnam argues that S implements the sequence ABABABA. Since both the automaton and the physical system are arbitrary, Putnam claims, the argument generalizes to any physical system and any (inputless and outputless) automaton. Putnam's argument goes as follows:

> Let the beginnings of the intervals during which S is to be in one of its stages A or B be t_1,t_2… t_n (in the example given, n = 7, and the times in question are t_1 = 12:00, t_2 = 12:01, t_3 = 12:02, t_4 = 12:03, t_5 = 12:04, t_6 = 12:05, t_7 = 12:06). The end of the real-time interval during which we wish S to "obey" this [computational description] we call t_{n+1} (=t_8 = 12:07, in our example). For each of the intervals t_i to t_{i+1}, i = 1, 2, …, n, define a (nonmaximal) interval state s_i which is the "region" in phase space consisting of all the maximal states … with $t_i \leq t < t_{i+1}$. (i.e., S is in s_i just in case S is in one of the maximal states in this "region.") Note that the system S is in s_1 from t_1 to t_2, in s_2 from t_2 to t_3, …, in s_n from t_n to t_{n+1}. (Left endpoint included in all cases but not the right — this is a convention to ensure the "machine" is in exactly one of the s_i at a given time.) …
>
> Define A = $s_1 \vee s_3 \vee s_5 \vee s_7$; B = $s_2 \vee s_4 \vee s_6$.
>
> Then, as is easily checked, S is in state A from t_1 to t_2, from t_3 to t_4, and from t_5 to t_6, and from t_7 to t_8, and in state B at all other times between t_1 and t_8. So S "has" the table[9] we specified, with the states A,B we just defined as the "realizations" of the states A,B described by the table. (Putnam 1988, 122–3, emphasis original)

To sum up, Putnam begins with an arbitrary physical system (e.g., an ordinary rock), slices up its continuous dynamics into arbitrary time intervals, and then aggregates the slices so that the aggregated slices correspond to arbitrary computational states and the system's dynamics correspond to an arbitrary sequence of computational states. He concludes that every physical system implements every inputless and outputless finite automaton. Allowing that inputless finite-state automata may be defined

[9] "Table" is Putnam's term for the transition rules that determine the computational state transitions of a finite-state automaton. He calls the physical state subspace s_i visited in the time interval t_i to t_{i+1} an "interval state", although it is not a properly defined physical state.

so that their initial states encode (finitely many) arbitrary digital inputs and their final states encode (finitely many) arbitrary digital outputs, Putnam's conclusion might in this sense be supposed to generalize to all finite-state automata with inputs and outputs.

The first thing to notice is that Putnam's construction seems to satisfy **S**, which requires that computational states correspond to distinct physical microstates of a system or disjunctions of physical microstates. To be sure, **S** requires that this hold for any trajectory of the system through state space, whereas Putnam defines his construction only for one trajectory. But the trajectory selected by Putnam is an arbitrary one, and every trajectory is presumed to be similarly "time stamped", so his construction generalizes to all possible dynamically supported trajectories.

By the same token, Putnam's construction seems to satisfy **P**, which requires that computational state transitions correspond to time evolutions of a system's physical microstates, possibly restricted to specific time intervals. Again, for **P** to actually be satisfied, Putnam's construction needs to be generalized to all trajectories of the system, but this alone is unproblematic. By satisfying **S** and **P**, Putnam's construction for inputless automata qualifies at least as a weak computational description in the present sense. We now address whether it can also satisfy **S'** and **P'**, as robust and strong computational descriptions require, and whether it does indeed generalize to include inputs and outputs that are themselves encoded in FSA states.

S' requires that computational states correspond to distinct physical microstates of a system, or to statistical states defined independently on disjoint regions of a system's state space. The computational states defined by Putnam do correspond to disjoint regions of a system's state space, but not to statistical states as we have defined them in Sec. 2(a)—as statistical states that encode nothing more about the system's dynamical history than is provided by the computational description. Putnam's construction fails in this respect, as we now argue.

In Putnam's construction, it is clearly not the case that every microstate belonging to a given computational state evolves from every microstate belonging to its predecessor computational state with the same probability, i.e., that statistical physical states can be unconditionally assigned to computational states. A microstate in the subspace s_2, for example, belongs to computational state B, which is always preceded by computational state A. But this microstate can only have evolved from microstates in subspace s_1 of computational state A; it could not have evolved from the microstates in subspaces s_3 and s_5 that also constitute A. This is a straightforward violation of **S'**.

This may seem to suggest that **S'** is overly restrictive. After all, Putnam's construction ensures that a microstate in computational state B (e.g. in s_2) evolves from a microstate in computational state A (e.g. in s_1) and that it will evolve into a microstate in computational state A (e.g. in s_3), thus respecting the state transitions of the FSA under consideration and perhaps seeming to satisfy the "spirit" of **S'**. However, the example FSA of Putnam's construction has the property that every computational state has a unique predecessor, and is far from arbitrary in this sense. His example need only be complicated slightly to more clearly reveal the failure to satisfy **S'** and the consequences of this failure. We illustrate

this with two example FSA that augment Putnam's example FSA with a third state and that include one computational state that has more than one predecessor.

First, consider an FSA with computational states A, B and C. A and B obey the same transition rules as in Putnam's example, but C is an initial state that cannot be "reached" by either A or B and always transitions to B. Over the time interval considered in Putnam's example, this FSA thus goes through the sequence CBABABA. State B now has two predecessors—state C and state A—which transition into B *at different times*. To say that the system is in a microstate belonging to subspace s_2 says not only that the system is in computational state B, but also that its predecessor was computational state C rather than computational state A; the microstate encodes more information about the system's dynamical history than is available in the computational description (computational states and transition rules). Second, consider a three-state FSA where A always transitions to B, B always transitions to C, and C always transitions to itself (i.e. C is a halting state). Suppose that over the time interval considered in Putnam's example, this FSA goes through the sequence ABCCCCC. Here, state C has two predecessors—state B and itself. Retaining Putnam's time intervals and associated state subspace labels, this system is in computational state C when its microstate belongs to the subspaces s_3-s_7. However, its predecessor computational state is identifiable as A if the microstate is in s_3 and as C if the microstate is in s_4-s_7. Here, physical microstates are again encoding more about the system's computational history than the system's present computational state.

Failure to satisfy **S'** in both spirit and letter, established above, is enough to preclude association of Putnam's construction with anything stronger than a weak computational description. It is instructive, however, to also consider potential satisfaction of **P'**.

P' requires that during the relevant time intervals, each physical microstate that corresponds to a given computational state evolve from (into) a microstate that corresponds to its predecessor (successor) computational state as specified by a set {(C,I)→C'} of deterministic state transition rules. Since Putnam's construction is specifically for inputless FSA, satisfaction of **P'** can only be tested for sets {C→C'} of deterministic state transition rules. If we want to separately consider the possibility of inputs encoded in initial states of otherwise "inputless" FSA, then we must also consider situations where multiple possibilities for initial FSA computational states are defined.

Putnam's construction can indeed accommodate computational state transitions for arbitrary inputless FSA, as mentioned above, and does satisfy **P'** for such FSA provided that there is only one possible initial state. Such FSA are, however, of a very restricted and trivial class.

If we wish to broaden this class just enough to accommodate multiple possibilities for initial states, and thus accommodate inputs in the only sense that FSA with no state branching can be regarded as accommodating inputs, we must associate distinct subspaces of the physical system's state space at the initial time (t_1 in Putnam's construction) with separate initial FSA computational states—one for each FSA input or input string. (Even **S** would require as much.) If the inputs encoded in this manner are to influence the FSA behavior—and thus have any meaning as FSA inputs—then the FSA must at later times be in computational states that depend upon the initial states and depend upon them in the right way.

This requires that we associate distinct subspaces of the physical system's state space at every computationally relevant time $t > t_1$ with each FSA computational state that, given the possibilities for initial states and the FSA table, could be accessed at that time. This alone is unproblematic and utterly conventional. However, it also requires that all microstates belonging to a particular state subspace at initial time t_1 evolve into the right state subspace at later times, necessitating the imposition of constraints on the system's dynamics. Without such constraints, nothing prevents two microstates associated with different initial computational states at t_1 from evolving into the same microstate at some computationally time $t_i > t_1$—even when the FSA transition rules would require that these initial computational states end up in different computational states at t_i. For the arbitrary dynamics of Putnam's construction, which admit temporally intersecting microstate trajectories, there are by definition no such dynamical constraints and **P'** cannot be satisfied for FSA with multiple initial states. No modified construction that includes dynamical constraints ensuring satisfaction of **P'** in such scenarios could be considered a variant of Putnam's, which is supposed to demonstrate that *arbitrary* dynamics of arbitrary physical systems implement arbitrary FSA.

In conclusion, Putnam's construction is supported only under weak computational descriptions of physical systems—it fails to satisfy both **S'** and **P'**. More specifically, while it satisfies **P'** for FSA that are inputless and outputless in the usual sense, it fails to satisfy **P'** for FSA with inputs encoded in their initial states and, more importantly, always fails to satisfy **S'**. Putnam's construction comes closest to satisfying both **S'** and **P'** for FSA with only one possible initial computational state and without any computational states that have more than one predecessor computational state—an exceedingly special and trivial subclass of FSA—but fails to satisfy **S'** even here.

One final remark on Putnam's construction is in order before we move on. The physical microstates that lie along any given trajectory in Putnam's construction are again distinguished from one another by their implicit "time stamp"—a signature of interaction with fields generated by an external "clock" system (e.g. the solar system). This time stamp renders other details of the microstate trajectories unimportant for the purposes of associating microstates with time intervals and thus with computational states in his construction. It is indeed reasonable to assume that every system is unavoidably bathed is such fields and necessarily interacts with them. It is not, however, clear that such fields provide the time stamp on the physical microstates that Putnam requires: the evolution of physical microstates is indeed influenced by externally applied fields, but particle microstates themselves are typically defined in terms of time-independent coordinates, velocities, etc. and not the forces acting upon them. In the above, we have given Putnam the benefit of the doubt and assumed that the microstates are indeed "time stamped" as he requires.

If one did not wish to accept that all microstates lying along every trajectory are distinguished by the external clock field, then there are two alternatives. First, if one wanted all microstates along every trajectory to be distinguished by a time stamp as in Putnam's construction but did not want to rely on external fields to provide it, one could just glue a stopwatch to Putnam's rock or any other system to build an internal clock into that system. Although this would not weaken Putnam's conclusion very much—it would still be radical to claim that a rock glued to a stopwatch implements every FSA—it would not immunize his argument from the objections we have advanced here. Alternatively, if one

wanted to abandon the requirement for time stamping of the physical microstates altogether, they could simply distinguish the microstates belonging to different computational states solely by standard state variables (e.g. coordinates and velocities). This comports with the conventional practice of mapping between computational states and physical states—with no temporal reference—assumed for real computing systems. Here, however, the association of microstate trajectories with sequences of computational states is no longer arbitrary. With no notion of time stamping, and under arbitrary dynamics, there is no prohibition against any given microstate being visited more than once along a dynamically supported trajectory (even if this is unlikely in all but the smallest physical systems). The non-cyclic behavior required for Putnam's argument—that any physical system implements any FSA—is thus incompatible with this more conventional view of computational-to-physical state mapping. Thus, neither of the above alternatives to Putnam's external clock allows his brand of unlimited PC to be supported by anything stronger than a weak computational description.

Searle's Wall and Hinckfuss's Pail

Searle's wall and Hinckfuss' pail contain no more resources than does Putnam's rock, so they fail to provide computational descriptions any stronger than those provided by Putnam's construction. Although these examples go beyond Putnam's in that they are purported to implement programs that depend on inputs—word processing and "a human program"—their random and unconstrained dynamics can only fail to produce the right dependencies between states and inputs exactly as in Putnam's construction**.** At best, one can try to interpret trajectories through the state space computationally as Putnam does for his rock, which, even with inputs encoded in initial computational states, cannot be achieved in a manner that satisfies both **S'** and **P'** as we have demonstrated above.[10]

A Falling Apple

A physical system with simple deterministic dynamics, such as an apple falling from a tree, may satisfy not only **P** but also **P'** without time-stamping of its physical states. The dynamics are such that there are no physical states—defined here by the position of the apple's center-of-mass and the velocity with which it is falling—that will be revisited as the apple falls to the ground, and these state variables alone distinguish the states. If segments of its state-space trajectory are labeled as computational states, it follows that, as **P'** requires, computational state transitions correspond to time evolutions of a system's physical microstates such that during the relevant time intervals each physical microstate that corresponds to a given computational state evolves from (into) a microstate that corresponds to its predecessor (successor) computational state as specified by the computational description. This only works if we are not looking for any dependence on inputs in the usual sense—the dynamics of the system are too simple to allow for that—although the inputs could be encoded in initial states by

[10] While Searle explicitly acknowledges that computation requires more than unconstrained dynamical evolution, he leaves the required constraints unspecified and employs an example—a wall—with random and essentially unconstrained microphysical dynamics. His key point is that, because computational descriptions are underdetermined by the microphysical dynamics, interpretation necessarily plays a role in using a physical system for computation. We agree that interpretation is necessary—if grossly insufficient—for such use. In this work we are addressing what else is required for sufficiency—specifically, the minimal requirements on a system's state transformations that would enable a user to give them meaningful computational interpretations.

dropping the apple from various lateral positions. But systems of this kind violate **S'**. Contrary to **S'**, all computational state transitions cannot "arrive at" or "depart from" an arbitrary physical microstate s belonging to computational state C with probability p(s,C) regardless of their source. Instead, the system goes through a specific sequence of microstates in accordance with its deterministic dynamics, so the "entry" and "departure" microstates for a computational state C can only be those unique microstates that the system is in at the beginning and end of the time interval associated with C. By violating **S'**, this would-be (Falling) Apple Computer is supported only under weak computational descriptions.

A Random System

A system with completely random transitions between microstates that are not time-stamped, in which each state transitions into some other state is selected completely at random, satisfies not only **S** but also **S'** by definition—the probability that the system goes into (or comes out or) any given state is independent of where it comes from (goes into). However, for this same reason that **S**' is satisfied, individual microstates will fail to always evolve from (into) predecessor (successor) microstates in a way that respects any deterministic computational state transition rule and **P'** is violated even for inputless FSA. (If state transitions depend on inputs, then the same successor computational state must be reached every time a given input is received while in a certain computational state, but then the state transitions cannot be random in the first place.) So, under some conditions, a completely random system can satisfy **S'** but fail to admit a robust computational description.

Chalmers's Clock-and-Dial System

David Chalmers (1996) offers a refinement of Putnam's construction to the effect that every physical system containing a clock and a dial implements every (inputless and outputless) finite-state automaton. By "clock," Chalmers means a subsystem that reliably transitions through a sequence of states c_1, c_2, …, regardless of environmental conditions. By "dial," Chalmers means a subsystem, with an arbitrary number of states d_1, d_2, …,, which reliably stays in any given state regardless of environmental conditions. This system implements every (inputless and outputless) automaton because for any initial state of the automaton, a dial state can be selected, and the sequence of computational states that corresponds to that initial state can be used to label the clock's sequence of states in the same way Putnam labels the microstates in his construction. With enough dial positions, there will be clock-dial state sequences available for mapping onto a sufficient number of computational state sequences that the full transition table of any FSA can be respected.

There are two differences between Putnam's construction and Chalmers's dial-and-clock system. First, the latter's states and state transitions are intrinsic to the system—they do not rely on interactions between the system and its environment. As a result, one need not rely on Putnam's Principle of Noncyclical Behavior to ensure that no physical state is visited more than once at computationally relevant times along any dynamically supported trajectory; the state variables associated with the effective "hand position" of the internal clock effectively provide an internal time-stamp that is built explicitly into the state specification. Second, Chalmers's system (unlike Putnam's) is explicitly defined

for all state space trajectories even with inputs encoded in initial states. Thus, Chalmers's system is defined more restrictively than Putnam's.

By adding the clock and dial, Chalmers solves three problems with Putnam's construction. First, the time stamping of microstates—required to ensure that no state is visited more than once along any dynamically supported trajectory—is unambiguously provided by the clock. Second, states and state transitions that are reflected in an FSA state table but that cannot all be visited in a given "run" of the FSA—some of which are necessarily neglected in Putnam's construction for such FSA—are captured through use of the dial. Third and finally, the clock and dial together ensure that distinct inputs encoded in initial FSA states can be assigned to distinct FSA state sequences that would result from any arbitrarily chosen FSA transition table, with the distinguishability of the dial states precluding temporal intersection of microstate trajectories (and thus violation of **P'**) for systems with inputs encoded in multiple possible initial states.

Chalmers's construction is still not as general as we might like, since inputs can only be encoded in the initial state of the clock-and-dial apparatus and thus can affect the evolution of the system's state only by being encoded in the system's state. But it does unambiguously enable satisfaction of **P'** for arbitrary FSA with inputs, even though the assignment of physical to computational states is otherwise just as ad hoc as it is in Putnam's construction. Regardless of which trajectory is followed, a given computational state evolves from (into) a microstate that corresponds to its predecessor (successor) computational state, as required by **P'**. This improves significantly on Putnam's construction in ways that suggest it can provide stronger support for unlimited PC.

However, while Chalmers's construction goes beyond Putnam's by satisfying **S** and **P'** even for FSA with inputs, it fails to satisfy **S'** and thus—like Putnam's—is supported only under weak computational descriptions. Computational states do correspond either to distinct physical microstates of a system or to statistical states defined independently on disjoint regions of a system's state space, as required by **S**'. As in Putnam's construction, however, the physical microstates belonging to the various computational states contain more information about the dynamical history of the system than do the computational states themselves.[11]

Chalmers's dial-and-clock construction thus provides a weak computational description at best. Other authors have offered variants of Chalmers's construction (Brown 2012, Scheutz 2012). They are more complex than Chalmers's original construction in that they employ many dials and clocks distributed over many subcomponents of a physical system. But the constraints they impose on the physical system are essentially the same. Therefore, they also satisfy only weak computational descriptions.

[11] Chalmers also offers a slightly modified construction for input-output FSAs. He claims that "every physical system with an input memory and a dial implements every FSA with the right input/output dependencies" (1996, 322), where an input memory is a subsystem that goes into a distinct state for every possible sequence of inputs. Chalmers does not explain how the correct outputs are supposed to come out at the end of a process that, as he defines is, does not depend on the input (or input memory state) in any way; in fact, in a later paper he points out that his dial-and-input-memory construction is mistaken (Chalmers 2012, 236).

Taking Stock

As we have seen, examples and constructions that motivate unlimited PC are supported only under weak computational descriptions—descriptions that set a low bar for "qualification" of physical states and/or processes as computational states and processes. We have shown explicitly why the descriptions that support them are weak, appealing to notions of representational faithfulness and computational fidelity. To conclude our discussion of unlimited PC, we review objections that have been raised in the literature and show how they are captured and made precise within our analysis.

One critique of unlimited PC is that the mappings between computational and physical microstates it relies on are illegitimate because they are constructed ex post facto (Copeland 1996). In other words, given these mappings, the work of generating successor computational states or outputs is not done by the physical system itself but by the person who constructs the mapping relation. We are now in a position to see exactly what this means. Weak computational descriptions may be weak because they violate **P'**, which means that the microphysical state transitions do not correspond to the computational dynamics. But weak computational descriptions may also be weak because they violate **S'**, which means they contain too much information—information about specific predecessor computational states of a physical microstate that is not included in the computational description.

Another critique of unlimited PC is that the mappings invoked by unlimited PC violate the counterfactual relations that must obtain between the computational states (Chalmers 1995, 1996, Copeland 1996). This observation gives rise to the counterfactual account of computation. We can now make more precise what is right about this objection. As we've seen, unlimited PC requires weak computational descriptions, and weak computational descriptions fail to support only and all of the right counterfactuals between computational states. Specifically, descriptions that violate **P'** fail to support the counterfactual relations encoded in the transition rules {(C,I)→C'} and descriptions that violate **S'** support counterfactuals not encoded in these rules. As a result, even descriptions that have been claimed to support the right counterfactuals—such as Chalmers's dial-and-clock construction (which satisfies **P'**)—can in fact fail to support the only and all of the right counterfactuals by violating **S'**. Descriptions that violate **S'** support counterfactuals they should *not* support—namely, counterfactuals from specific microstates corresponding to one and the same computational state to specific predecessor states, counterfactuals that are not supported by the transition rules {(C,I)→C'}.

Another popular response to unlimited pancomputationalism is that its mappings fail to construct an isomorphism between the causal structure of the physical system and the state transitions specified by the computational description, and non-causal mappings are illegitimate (Chrisley 1995, Chalmers 1995, 1996, 2011, Scheutz 1999, 2001). This objection gives rise to the causal account of computation, according to which acceptable mappings must respect the causal structure of a system. Indeed, weak descriptions either fail to mirror the causal structure encoded in the computational state transitions (if it violates **P'**), or it contains too much causal structure (if it violates **S'**). As we have argued above, in order to capture all and only the causal structure of a system that is encoded in the computational state transitions, a description must satisfy both **S'** and **P'**—that is, it must be robust. Thus, robustness is a more precise and adequate replacement of the causal account of computation.

Yet another response to unlimited pancomputationalism is implicitly given by Godfrey-Smith (2009). Although Godfrey-Smith is primarily concerned with functionalism as opposed to computation per se, his argument is relevant here. Godfrey-Smith argues that for a mapping to constitute a genuine implementation, the physical microstates that are clustered together (to constitute a given computational state) must be physically similar to one another—there cannot be arbitrary groupings of arbitrarily different physical states. Godfrey-Smith suggests that his similarity restriction on legitimate mappings may be complemented by the kind of causal and localization restrictions proposed by Chalmers (1996). Our faithfulness and fidelity conditions quantify and make precise the similarity that is needed to make Godfrey-Smith's argument go through. The similarity between microstates must be such as to satisfy **P'**, but the microstates must not be so similar as to violate **S'**. Thus, our fidelity and faithfulness conditions make more precise and capture what is right about the counterfactual, causal, and similarity-based accounts of computation.

In summary, we have offered a more precise account of where unlimited PC goes astray. Unlimited PC relies on mappings between computational and physical descriptions that lack faithfulness, fidelity, or both. We can now tackle limited PC.

3(b). Limited PC

The weak version of pancomputationalism is limited PC.

Limited PC Every physical system objectively performs at least one computation.

Limited PC is weaker than its unlimited namesake. It holds that every physical system performs at least one computation, which may be labeled in different ways and may encode other (similarly or less complex) computations, or a small number of nonequivalent computations that satisfy appropriate conditions. Exponents of limited PC (e.g., Chalmers 1996, 331, Scheutz 1999, 191) usually reject unlimited PC and maintain that although every physical system performs some computation, which computation is performed by which physical system is an objective matter that depends on the properties of the system such as its causal structure at an appropriate level of abstraction.

One reason given for limited PC is that everything has causal structure. According to some, computation is the causal structure of physical processes at some level of abstraction (Chrisley 1995, Chalmers 1995, 1996, Scheutz 1999, 2001). If every physical system has causal structure, it follows that every physical system performs the computation constituted by its causal structure. This is *causal* limited PC.

Some people reject the notion of causation as non-fundamental and dispensable from fundamental physics (e.g., Norton 2003). But qualms about causation are not a reason against limited PC. Causation skeptics can recover an analogue of causal pancomputationalism by formulating an argument for limited PC in terms they like—e.g., in terms of the dynamical properties of physical systems.

Another alleged reason for limited PC is that every physical state carries information combined with the popular view that computation is information-processing. It follows that every physical system performs the computations constituted by the manipulation of its information-carrying states (cf. Shagrir 2006, Milkowski 2013). The view that computation is information-processing remains controversial.[12]

Unlike unlimited PC, limited PC does not trivialize the claim that a physical system is computational completely. Different systems generally have different objective properties; thus, according to limited pancomputationalism, different systems generally perform different computations. Computer scientists and engineers would overwhelmingly agree with this aspect of limited PC and would go even further, taking it for granted that relatively few physical systems—mostly those of their deliberate creation—perform computations. By contrast, limited PC maintains that the digital computers created by computer scientists and engineers through enormous investments of ingenuity and labor perform computations in the same sense in which rocks, hurricanes, and planetary systems do—because they all have causal structure or because they all process information.

Limited PC is Supported Only Under Robust Computational Descriptions (At Best)

We will now argue that limited PC is supported only under robust computational descriptions because it requires satisfaction of **S'** and **P'** but cannot satisfy **U**. Before showing this, we need to set aside a trivial version of limited PC.

According to trivial limited PC, every physical system objectively performs some computation no matter how trivial that computation may be. For example, consider an ordinary toaster initially holding an untoasted piece of bread. Take the slider to be one input ("up" encoding 0 and "down" encoding 1), the electrical plug to be another input ("unplugged" encoding 0 and "plugged in" encoding 1), and the state of the bread to be the output ("untoasted" for 0 and "toasted" for 1), all defined at "computationally relevant" times. The toaster computes the AND function – or any of the other two-input, one-output Boolean function that can be obtained by relabeling the inputs and outputs of the AND function – since the slider has to be depressed and the toaster has to be plugged in if toasted bread is to result. But the two-input AND is a trivial computation with only two inputs and one output. If this kind of trivial computation is accepted, then limited PC is supported even under strong computational descriptions at least for any inputs and outputs that can be prepared and observed by users.

This being said, we will set trivial limited PC aside. Claims that, in the above sense, a toaster implements the AND function, a switch implements the NOT function, a coin sitting on a table implements the identity function — that any system implements some function of similarly low complexity— would be unimpressive, even if they held up under the harshest scrutiny. We should not be surprised that processes are to be found in our complex universe with causal structures coinciding with those of elementary, abstract mappings that we call computational primitives, and should not make too much of it. We could find many examples of common objects that objectively implement trivial computational

[12] The present authors' perspectives on this question are provided in (Anderson 2016b) and (Piccinini 2015).

primitives (like the toaster AND gate), but would be much harder pressed to identify objects that implement nontrivial computations (like a tree stump or garden trowel that performs image compression). From now on, we thus consider only nontrivial versions of limited PC, according to which every physical system performs a limited number of *nontrivial* computations.

We will now argue that (nontrivial) limited PC violates **U**; therefore, it requires computational descriptions that are at most robust (as opposed to strong). We begin by assuming the opposite of our conclusion: an arbitrary physical system computes nontrivial function *f*: I -> O so that the system may be used to evaluate *f* for arguments of an agent's choosing. We also assume that, if we can identify a nontrivial function *f*: I -> O computed by the system, we can then construct a robust computational description of the system, that is, a computational description that satisfies **S'** and **P'**.

These assumptions are highly nontrivial. First, satisfying both **S'** and **P'** requires specific properties that go beyond simply possessing causal structure or processing physical information. Thus, it takes more than the standard reasons given by supporters of limited PC to identify objective computations in any physical system. Second, that an arbitrary physical system performs nontrivial computations is dubious and remains to be shown. Thus, we are far from assuming that limited PC is true. But we need to assume that both **S'** and **P'** are satisfied so that the system's putative computations are "objective" in the sense of having robust physical implementations. We aim to show that even an arbitrary system that satisfies these assumptions still fails to satisfy **U** and thus its computations are not usable.

Select an arbitrary physical system S. We wish to identify the function *f* computed by S. To do that, we need to identify S's inputs and outputs. As we pointed out above, inputs may be identified either with influences that enter a system as it is evolving or with "initial" states. Similarly, outputs may be either signals that exit the system as it evolves or "final" states. We will now list challenges that make it hard or impossible to identify a strong computational description (i.e., one that satisfies **U** and therefore makes the computation usable).

A first challenge is that all relevant features of a physical system (states, external influences, and signals from the system) can be described at different levels of granularity. They can be described at different temporal and spatial scales, more finely or more coarsely. So, no matter how we choose the inputs and outputs, there are indefinitely many possible descriptions of the inputs and outputs and, corresponding to those, indefinitely many functions putatively computed by the system.

The cleanest solution to this problem would be to identify the fundamental physical level (if there is one) and define inputs and outputs at that level. This is the solution pursued by ontic PC (more on this in Section 3(c)). But this solution won't work for present purposes for two reasons. First, we don't know what the fundamental physical level is. Second, given our current technology we have no direct way to manipulate the fundamental physical level to select arbitrary values on which to evaluate *f* and observe the results of the putative computation. Therefore, we need to operate at a nonfundamental physical level. But there are many of those, and it's not clear which one we should choose. Let's pick an arbitrary level L. From now on, the function putatively computed by our system S is relative to level L. (At levels different from L, the system may compute different functions.)

A second challenge is that most systems are out of our reach. They are too big, too small, too distant in space, too far in the past or future, or insufficiently understood for us to observe and manipulate their inputs and outputs, which we must do in order to evaluate the function they putatively compute on arguments of our choosing. Therefore, we need to restrict our attention to systems that are within our reach.

A third challenge is that there are indefinitely many computational formalisms which we could use to define *f*. Any given formalism (finite-state automata, cellular automata, register machines, etc.) gives rise to different computational descriptions of *f*. For present purposes, we will stick with finite-state automata as our canonical computational description.

A fourth challenge is that in an arbitrary physical system, there is no well-defined distinction between computationally *relevant* degrees of freedom—those constituting the inputs, internal states, and outputs of the computation—and computationally *irrelevant* degrees of freedom, including those pertaining to the inflow of energy into the system, outflow of heat from the system, and any other arbitrary influences on the system and its dynamics. For instance, consider a falling leaf again. Are the air currents that push it around inputs to a computational system or random disturbances to its computational dynamics? Depending on how we answer this question, we end up with radically different computational descriptions. And regardless of how we answer this question, air currents are probably too disorderly to be captured in a perspicuous computational description, such that it is clear which argument of which function is being computed by the system and which step of the computation is being performed at any given time. A clean way around this challenge is to take the whole universe to be the computing system (more on this below). When only a subset of the universe is considered, however, the lack of a clean boundary between an arbitrary system's putative computational structure and other factors is a serious challenge.

This leads to our fifth and perhaps greatest challenge: (nontrivial) dynamical descriptions of arbitrary physical systems are not exact but approximate in a number of ways. To begin with, the microstates of a system can be measured and specified only with finite precision, so that there is a gap between what a microphysical description says and what the system does. Secondly, some aspect of the system's dynamics may be unknown, which enlarges the gap. Thirdly, including every known factor in the microphysical description of an arbitrary system (from which the computational description is to be constructed) typically makes the model mathematically or computationally intractable. Because of this, typical microphysical descriptions incorporate simplifications and idealizations, which may further enlarge the gap between what the microphysical description says and what the system does. Fourthly, accuracy in representing the dynamics of a system requires computational resources. The more accuracy is desired, the more resources are needed. Since computational resources are always finite, this introduces a further gap between what a microphysical description says and what the system does. Fifth, typical deterministic dynamics are nonlinear, and typical nonlinear dynamics are so sensitive that the system's actual dynamical evolution diverges exponentially from any dynamical description based on a finite specification of the system.

The approximate nature of ordinary dynamical descriptions, in combination with the lack of any sharp boundary between putative computational structure and external influences on an arbitrary (nontrivial) system, have a dire consequence: even if we succeeded in constructing a robust computational description of an arbitrary physical system as computing nontrivial function *f*, it is unlikely that we would be in a position to use the system to evaluate *f* for arguments of our choosing with any reliability. This is because, even if we could observe or prepare the system as being in a desired initial state I, we have no guarantee that the gap between our description and the actual behavior of the system, plus any other disturbances, would lead the system to yield the desired value O.

In conclusion, nontrivial limited PC is less compelling than it may have seemed. Even if we set aside the difficulties with identifying nontrivial robust computational descriptions of an arbitrary physical system such that the system computes function *f*, in general it is extremely unlikely that we could ever use the system to evaluate *f* for desired arguments. It takes a lot of careful regimentation for a system to let a user evaluate a function reliably. It takes regimentation in how the microstates are clustered, in how they transition over time, as well as in shielding the relevant state transitions from external disturbances. Most systems lack this regimentation. It is unlikely that systems that are not evolved or designed and built for computation will possess the exquisitely fine-tuned regimented structure that would allow them to perform nontrivial computations reliably. This is not to argue against the possibility that some artificial or natural systems—systems not commonly regarded as "computational"—may indeed possess this regimentation, which workers in the fields of unconventional and natural computation seek to discover and even utilize. But the nontrivial limited pancomputationalist's burden is to show that any sufficiently complex physical system performs one or more nontrivial computations. This burden has not been met.

Universe as a Computing System within Limited PC

Further questions arise about the strength of supporting computational descriptions for limited PC when we consider the notion that the whole universe computes. This notion is natural for ontic pancomputationalists, as discussed in Sec. 3(c), but the notion can and should be considered independently of ontic PC since all forms of pancomputationalism are supposed to apply to all physical systems. We could do this by taking those microstates of the universe that have been visited so far to be computational states, the transitions between these states to have been governed by the evolution of the universe, and the "function" computed by the universe to be its own evolution. Such a claim would belong to limited PC, but not unlimited PC (since only one computation is claimed for the system) and not ontic PC (since a mapping between physical and computational states is involved, rather than the claim that they are one and the same or are on an equal footing). We now consider computational states, computational processes, and usability for the universe-as-a-computing-system, and assess the implications for limited PC. For consistency, this discussion assumes a classical, causal, globally closed, deterministically evolving universe. (We will discuss the quantum mechanical universe in the next section.)

Selection of the past microstates of the universe as the computational states of the "universe-as-a-computing system" is the only objective choice for these states, as a computation constructed from the sequence of these states that has been carved out in the universe's past history would encode any computation constructed from any disjunction or coarse graining of these states. Thus, no statistics are involved in mapping statistical physical states to computational states and the stronger condition **S'** on computational states is trivially supported.

The stronger condition **P'** on computational processes is also supported, but also trivially. Assuming that there is one universe with a unique history, no universe microstate - and thus no computational state - has been visited more than once. Construction of any computation from this sequence of states can thus have no branching, and is analogous to a non-cyclic, inputless FSA. **P'** is easily satisfied for this case.

One could argue that, if the universe is governed by causal physical laws, we need not be restricted to a computational description constructed solely from the universe's history. We could use these physical laws to build more complex computational descriptions of the universe-as-a-computing-system that accommodate the possibility of inputs and computational branching, with causal properties and counterfactual robustness underwritten by regularities described by these laws. It is not clear, however, what could possibly count as an "input" to the universe as a whole, at least with the common conception of an input as an external influence. Any input that is somehow "internal" to the system is necessarily reflected in the complete specification of the system's state, and everything in the universe is by definition internal to the universe. Paradoxically, if the computation implemented throughout the universe's history is indeed without inputs, it is in this sense a trivial (if long) computation.

The usability criterion also takes on a different light when the universe as a whole is considered as a computing system. There are two reasons for this. First, it is not clear that an "agent" can be defined that is not internal to the universe. While we are living proof that internal agents do exist, our relevance to the universe-as-a-computing-system is limited since the computing systems we use are subsystems of the universe (such as laptop computers) and not the universe as a whole. Second, even if an external agent can be defined, it is not clear what it would mean to say that they use the universe-as-a-computing-system to evaluate the universe-function for an argument of their choice. If such a function could, in principle, be identified, and if we allow that an argument was encoded in the initial state of the universe (since there are no external influences), then the function is (or is being) evaluated for one and only one argument. There is no chance of evaluating it for another argument without reenacting the entire history of the universe with different initial conditions.

In conclusion, consideration of the universe as a computing system upholds the above conclusion - that the robustness condition (**S'** and **P'**), but not the usability condition **U**, are supported by limited PC - but only in the sense that **S'** and **P'** are trivially satisfied and **U** goes unsatisfied because it is of dubious applicability in this context.

3(c). Ontic PC

Some authors—mostly physicists—argue that the physical universe is *fundamentally* computational. It's not just that the universe itself is a computing system and everything in it is a computing system too, which is limited PC. It's that there is a fundamental level of physical reality at which the one and only fundamental computation performed by each physical system can be identified. In addition, that fundamental computation is all there is to the nature of a physical system.

An alternative formulation is that information is what makes up the universe. Proponents of the first formulation typically think of computational states as bearers of information, and proponents of the second formulation typically think of information dynamics as computations. Since typical proponents of ontic PC think of computation as information processing and vice versa, then, the two formulations are roughly interchangeable for present purposes. We will continue to use the computational formulation of ontic PC—as opposed to the informational formulation—and define:

> **Ontic PC** Every physical system objectively performs one computation, which exhausts the nature of the physical system.

Ontic PC includes both an empirical claim and a metaphysical one. The empirical claim is that fundamental physical magnitudes and their state transitions are exactly and exhaustively describable by an appropriate computational formalism—without the approximations that are ubiquitous in standard computational descriptions of physical systems. This empirical claim takes different shapes depending on which computational formalism is assumed to describe the universe exactly. We will briefly discuss the two most widely held forms of ontic PC are "digital ontic PC" based on cellular automata— a classical computational formalism—and "quantum ontic PC" rooted in quantum computing. The metaphysical claim of ontic PC is that computation is what makes up the physical universe. This point is sometimes made by saying that, at the most fundamental physical level, there are brute differences between states—nothing more need or can be said about the nature of the states. This claim requires elucidation. Before we get to that, we will discuss the empirical component of ontic PC.

Ontic PC sidesteps the requirements of computational description. Specifically, by reducing the physical to the computational, ontic PC resists rigorous classification via the criteria we use in this work to gauge the strength of the underlying computational description required for its support. These criteria are germane to presumed distinctions between physical description and computational description that are not recognized in ontic PC. For this reason, rather different considerations must be brought to bear on detailed evaluation of ontic PC, and we provide such an evaluation in (Piccinini and Anderson 2017). That said, we note that ontic PC—in both the classical-digital and quantum —is most closely associated with limited PC: the computational descriptions posited by ontic PC are stronger than weak computational descriptions because they objectively assign a unique computational trajectory to every possible dynamical evolution of the universe, but, by erasing the distinction between a computing system and an agent that would employ it to evaluate a function, they cannot satisfy the usability criterion required of strong descriptions.

4. Conclusion

In this work, we have characterized three different versions of pancomputationalism (PC)—unlimited PC, limited PC, and ontic PC—in terms of the nature and strength of the computational descriptions required to support them. To this end, we distinguished three classes of computational description that reflect distinct views of the relationships between physical microstates and computational states, the relationship between physical dynamics and computational state transitions, and the usability of physical systems for computation by agents.

Weak computational descriptions admit mappings between computational states and physical microstates that lack robustness, i.e. mappings that allow physical microstates to embody either more or less information about the computational history (i.e. previous computational states) of a system than is embodied in the computational states themselves.

Robust computational descriptions require that the amount of information about a system's dynamical history that is encoded in the physical microstates is identical to that encoded in the computational states, but not that the system is usable by an external agent to evaluate a function for arguments of the agent's choice.

Finally, strong computational descriptions are robust and are usable by an agent to evaluate a function for arguments of the agent's choice. Needless to say, strong computational descriptions apply to the digital computing devices that we buy, sell, and use every day. Strong computational descriptions are the gold standard.

Unlimited PC claims that every physical system performs a large number of nonequivalent computations. This claim can be supported only under weak computational descriptions.

Limited PC claims that every physical system performs a limited number of computations that satisfy some condition depending on the objective properties of the system. While limited PC may be supported under robust or even strong computational descriptions for systems that perform trivial computations, limited PC has not been established for any nontrivial computations. Furthermore, even if limited PC were established for nontrivial computations, it would be supported at best by robust (as opposed to strong) computational descriptions.

Ontic PC is a special version of limited PC. It claims that every physical system is at bottom just a computational system. This means that the nature of every physical system is exhausted by the computation it performs at its most fundamental physical level. Ontic PC thus cannot be classified according to scheme employed in this work—i.e. by the strength of the computational description that it supports—so we have provided a separate analysis elsewhere (Piccinini and Anderson 2017). The underlying requirement for its acceptance is essentially acceptance that two systems that are formally or observationally equivalent in some (though not all) respects are the same. Even more is required for acceptance of universal ontic PC, which views the universe as a computer simulating its own behavior.

This erases the distinction between a simulator and its simulated target system, requiring that we accept a system as a simulation of itself.

Strong computational descriptions, which are the gold standard, do not support any version of PC that we considered here or that we are otherwise aware.

Perhaps the best way to illustrate our conclusions is to consider what it might mean to associate time evolution of the state of a physical system with performance of a specified computation *f*. Possible meanings include "that particular evolution of the system was consistent with having computed *f*," "the system's evolution will always objectively compute *f*," or "the system's evolution can always be harnessed by an agent to perform *f*," where computation of *f* refers to a full mapping from a set of physical input (or initial) states to physical output (or final) states that has a structure similar to the structure of the function *f*.

The weak computational description we have associated with unlimited PC would sanction only something like this: "For any given sequence of physical microstates visited in the arbitrary time evolution of any sufficiently complex system (e.g. a rock), there exists an assignment of physical microstates to computational states such that time evolution of the microstates is consistent with the system *having performed* the computation *f*." This is not very impressive, in part because of the freedom to specify computational states *ex post facto* and without constraint, and is certainly less sensational than unlimited pancomputationalist headlines announcing that rocks and walls have the capacity to compute everything or that pails of water are occasionally computationally equivalent to a conscious mind.

The robust computational description we have associated with limited PC is stronger than this, in that it requires regularities in the behavior of physical systems that allow us to state that – for computational states specified *ex ante* — the system "always objectively performs" computation of some function *f*. For trivial limited PC, however, *f* is too simple to impress; the recognition that a toaster reliably implements the AND function is unsurprising and will not prompt Intel to begin building computers from kitchen appliances. Even the trivial limited PC that would regard the "universe as a computing system" is unimpressive, since it corresponds to a single trial of a system that is in the process of computing an unknown *f* for one and only one input (initial condition) and can never be used again. The dearth of even toy examples illustrating nontrivial limited PC – something like a doorstop that objectively performs matrix multiplication or a spatula with the capacity for natural language processing – suggests that nontrivial limited PC is implausible. It is simply exceedingly unlikely that systems that are neither designed nor evolved for computation (like iPads and possibly brains) will possess regimented structure that would give them the capacity to perform nontrivial computations. The existence of particular physical systems that may unexpectedly possess nontrivial computational capacities when harnessed in specific ways, such as those studied in the fields of natural and unconventional computing, does not help the pancomputationalist, who asserts that all physical systems possess computational capacities without the need for us to constrain or harness them in any way.

Finally, although ontic PC is of a special nature that frees it from some constraints used to analyze unlimited and non-ontic versions of limited PC – and may be rich in insights when throttled back to a more metaphorical form of PC – its dramatic pronouncements lose much of their force when the required hypotheses or redefinitions of familiar terms are explicitly acknowledged. A claim that "the universe is a computing system" is not particularly impressive if it simply acknowledges that the freely evolving physical universe qualifies as a computing system under a definition of "computing system" that admits the freely evolving physical universe.

References

Anderson, N. G. (2010). On the physical implementation of logical transformations: Generalized L-machines. *Theoretical Computer Science*, *411*(48), 4179-4199.

Anderson, N. G. (2016a). Information Processing Artifacts. Presented at the Twenty-Fifth Biennial Meeting of the Philosophy of Science Association (Poster Forum), Atlanta, November 3-5, 2016. Manuscript in preparation.

Anderson, N. G. (2016b). Information as a physical quantity. Under review. Preprint available: DOI 10.5072/FK2SB42W08.

Bennett, C. H. (2003). Notes on Landauer's principle, reversible computation, and Maxwell's Demon. *Studies In History and Philosophy of Science Part B: Studies In History and Philosophy of Modern Physics*, *34*(3), 501-510.

Block, N. (1978). "Troubles with Functionalism," in *Perception and Cognition: Issues in the Foundations of Psychology*, C. W. Savage (ed.), Minneapolis, University of Minnesota Press, pp. 261-325.

Brown, C. (2012). Combinatorial-state automata and models of computation. *Journal of Cognitive Science*, *13*(1), 51.

Chrisley, R. L. (1995). Why everything doesn't realize every computation. *Minds and Machines*, *4*(4), 403-420.

Chalmers, D. J. (1995). "On Implementing a Computation," *Minds and Machines*, 4: 391–402.

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

Chalmers, D. J. (2011). A Computational Foundation for the Study of Cognition. *Journal of Cognitive Science*, 12(4), 323-357.

Chalmers, D. (2012). The varieties of computation: A reply. *The Journal of Cognitive Science*, *13*, 211-248.

Copeland, B. J. (1996). What is computation? *Synthese*, *108*(3), 335-359.

Ganesh, N., & Anderson, N. G. (2013). Irreversibility and dissipation in finite-state automata. *Physics Letters A*, *377*(45), 3266-3271.

Godfrey-Smith, P. (2009). Triviality arguments against functionalism. *Philosophical Studies*, *145*(2), 273-295.

Houkes, W., & Vermaas, P. E. (2010). *Technical functions: On the use and design of artefacts* (Vol. 1). Springer Science & Business Media.

Hughes, J. (2009). An artifact is to use: an introduction to instrumental functions. *Synthese*, *168*(1), 179-199.

Klein, C. (2008). Dispositional implementation solves the superfluous structure problem. *Synthese*, *165*(1), 141-153.

Ladyman, J., Presnell, S., Short, A. J., & Groisman, B. (2007). The connection between logical and thermodynamic irreversibility. *Studies In History and Philosophy of Science Part B: Studies In History and Philosophy of Modern Physics*, *38*(1), 58-79.

Ladyman, J. (2009). What does it mean to say that a physical system implements a computation?. *Theoretical Computer Science*, *410*(4), 376-383.

Landauer, R. (1961). Irreversibility and heat generation in the computing process. *IBM journal of research and development*, *5*(3), 183-191.

Lycan, W. G. (1981). Form, function, and feel. *The Journal of Philosophy*, 24-50.

Maudlin, T. (1989). Computation and consciousness. *The Journal of Philosophy*, 407-432.

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

Mills, J. W. (2008). The nature of the extended analog computer. *Physica D: Nonlinear Phenomena*, *237*(9), 1235-1256.

Norton, J. D. (2003). Causation as folk science. *Philosophers' Imprint*, 3(4).

Piccinini, G. (2010). "The Mind as Neural Software? Understanding Functionalism, Computationalism, and Computational Functionalism." *Philosophy and Phenomenological Research*, 81(2): 269-311.

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

Piccinini, G. and N. G. Anderson (2017), "Ontic Pancomputationalism ," in *Computational Perspectives on Physics, Physical Perspectives on Computation*, S. Fletcher and M. Cuffaro (eds.), Cambridge University Press, (forthcoming).

Pour-El, M. B. (1974). Abstract computability and its relation to the general purpose analog computer (some connections between logic, differential equations and analog computers). *Transactions of the American Mathematical Society*, *199*, 1-28.

Putnam, H. (1988). *Representation and reality* (Vol. 454). Cambridge, MA: MIT press.

Rescorla, M. (2014). A theory of computational implementation. *Synthese*, *191*(6), 1277-1307.

Rubel, L. A. (1989). Digital simulation of analog computation and Church's thesis. *The Journal of Symbolic Logic*, *54*(03), 1011-1017.

Scheutz, M. (1999). When physical systems realize functions... *Minds and Machines*, *9*(2), 161-196.

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

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.

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

Shagrir, O. (2006). Why we view the brain as a computer. *Synthese*, *153*(3), 393-416.