Minds and Machines (2019) 29:193–225
https://doi.org/10.1007/s11023-019-09491-w

GENERAL DISCUSSION

Information Processing Artifacts

Neal G. Anderson[1]

Received: 31 August 2018 / Accepted: 16 January 2019 / Published online: 28 January 2019

Abstract
What is a computer? What distinguishes computers from other artificial or natural systems with alleged computational capacities? What does use of a physical system for computation entail, and what distinguishes such use from otherwise identical transformation of that same system when it is not so used? This paper addresses such questions through a theory of information processing artifacts (IPAs), the class of technical artifacts with physical capacities that enable agents to use them as means to their computational ends. Function ascription, use plan requirements, malfunction, and efficacy of IPAs are all addressed in this theory, with emphasis on artifacts that can be used—reliably or otherwise—for digital computation. By explicitly distinguishing physically grounded computational capacities from user-ascribed computational functions, and by recognizing the distinct roles of each for the implementation of computations in artifacts, this theory clearly distinguishes the use of physical systems for computation from the transformations of physical system states that enable such use. As such, it provides a rigorous basis for distinguishing "computers" from other artificial and natural systems—a distinction whose nature and legitimacy faces ever-evolving challenges from multiple disciplines. This theory, and the associated "instrumental" view of computation in artifacts, naturally accommodates the openminded but scrupulous consideration of radically unconventional physical systems as potential substrates for future computers.

Keywords Computers · Technical artifacts · Artifact functions · Physical information · Physical computation · Instrumental computation · Unconventional computation · Natural computation · Pancomputationalism

✉ Neal G. Anderson
anderson@ecs.umass.edu

[1] Department of Electrical and Computer Engineering, University of Massachusetts Amherst, Amherst, MA 01003-9292, USA

1 Introduction

Artifacts are distinguished from other physical objects by their usability. Whether created, evolved, or appropriated for a particular purpose, artifacts possess identifiable physical capacities that can be harnessed by agents through proper execution of prescribed use plans (Vermaas and Houkes 2006a, b; Hughes 2009; Houkes and Vermaas 2010). Manufactured digital devices—the signature technical artifacts of our time—naturally admit such a characterization: their computational capacities are deliberately "engineered in" and are user accessible by design. Such artifacts—*digital computing artifacts* (DCAs)—are obviously a species of what we here call *information processing artifacts* (IPAs), technical artifacts with physical capacities that allow agents to use them as means to their information processing ends.

Much less obvious are the general criteria that a physical system must satisfy to qualify as an IPA, or, more specifically, as a computing artifact—criteria that would unambiguously distinguish computers from other physical systems. Computers would seem obviously to be distinct in that they are used by agents to implement particular computations. If this is what distinguishes computers from other physical systems and artifacts, then they must be distinct in at least one two senses: they are distinct "intrinsically" because their dynamics unambiguously execute particular computations independent of agents, and/or they are distinct "extrinsically" because they are unambiguously used by agents to execute particular computations.

It is not at all clear that physical systems can be taken to execute particular computations in the intrinsic sense. The dynamics of real systems—systems interacting with their environments—evolve their physical states in ways that rule out any straightforward correspondence between physical state transformations and the computational state mappings that define *any particular* computation. The correspondence is of course close in conventional electronic computing machines, but can be far less so—by nature or by design—in the unconventional and natural systems increasingly under investigation as potential substrates for future computers. In any case, even where dynamics are such that there *is* a close correspondence between physical and computational state mappings, it remains to establish what is inherently computational about physical dynamics independent of an externally imposed computational interpretation. Either a purely dynamical hallmark of computation must be identified, or—if external interpretation is to play an essential role—there must be a clear sense in which physical dynamics can be intrinsically computational while requiring externally imposed computational interpretations. If there is such a sense—if externally imposed interpretations are admissible—it then remains to establish how the inherent subjectivity of computational interpretation does not imply, as the pancomputationalist would claim,[1] that any physical dynamics can trivially be taken to implement any computation.

[1] See Piccinini (2015, pp. 51–73), Anderson and Piccinini (2017), and Piccinini and Anderson (2018) for critical evaluation of pancomputationalist claims and constructions, including those discussed by Putnam (1991), Searle (1992), Chalmers (1996), and others.

It is also not entirely clear how computers are distinct from other physical systems in the extrinsic sense. The characterization of use presents challenges of its own, requiring that use of a system for computation be clearly distinguished from the otherwise identical physical transformation of that same system when it is not so used. If such a distinction exists, and if use for computation is required for physical systems to count as computers, then computers do not execute computations when they are not used by agents for computation—even when they evolve exactly as they do when they *are* used for this purpose. Can this be?

The distinguishing of computers from other physical systems thus faces challenges on multiple fronts, even if through the seemingly obvious recognition that they are artifacts used by agents to implement specific computations. In this paper, I offer a theory of IPAs and DCAs that addresses these challenges. This theory regards the execution of computations as functional goals of agents and regards computers as technical artifacts used by agents as means to achieve these goals.[2] By distinguishing computational capacity from computational function, and display of computational capacity from execution of a computational function, this theory clarifies what it means for agents to use physical systems to perform digital computations—reliably or otherwise—and what distinguishes computers from other artificial and natural systems. My objectives are to articulate this theory in detail, to show how this theory meets the above challenges, and to characterize the view of computation—hereafter "instrumental computation"—that underlies this theory. This view contrasts sharply with the notion of "intrinsic computation" in physical systems, in that it takes computation to be more than appropriately structured patterns of state transformations in physical objects. Instrumental computation further requires agents, without whom there are no computational goals, no artifacts with computational functions, and thus no computations.

Several remarks are in order at the outset.

– I will focus primarily on digital computing artifacts in this work. Specifically, I will focus on artifacts used to evaluate deterministic logical functions L for inputs of a user's choosing, which I will call L-IPAs. While L-IPAs belong to a restricted subclass of IPAs (and even of DCAs), any artificial or natural system that can be used to evaluate an arbitrary complex logical function L for any input in this function's domain is an L-IPA. The capacities of a system required for its use as an L-IPA are captured, at the right level of generality for the purposes of this work, by Ladyman's L-machines (Ladyman et al. 2007; Ladyman 2009) in idealized scenarios and by their noisy, quantum generalizations (Anderson 2010) in more realistic scenarios. Analogous accounts of capacities for realization of more sophisticated digital computing structures, such as the general physical characterization of finite state automata of Ganesh and Anderson (2013), could

[2] Note that Turner (2018) has articulated a philosophy of computer science that regards both computing machines and programs as technical artifacts, and which includes a sketch of "logical machines" that is similar in spirit to the theory of digital computing artifacts developed in detail in this work.

be used to extend the present theory to subclasses of DCAs that go beyond *L*-IPAs.

- The notion of function that I adopt and extend to computational functions is that of an instrumental artifact function introduced by Hughes (Hughes 2009). This notion of function[3] emphasizes instrumental user knowledge over explanatory knowledge, which is entirely appropriate in computational contexts. Digital computation, as practiced routinely by billions of users, overwhelmingly involves users with substantial instrumental knowledge of how to use their digital devices to accomplish various tasks but little or no explanatory knowledge of what gives rise to the computational capacities of these devices. There even exist systems with computational capacities that cannot be accounted for by their creators—systems with computational capacities rooted in physical structures that are known to no one—but that can demonstrably be used to execute nontrivial computations (e.g. Vissol-Gaudin et al. 2017).
- Because the computational functions of this work are instrumental artifact functions, they apply exclusively to artifacts. Instrumental artifact functions presume—indeed require—the existence of agents who have functional goals, ascribe functions to artifacts, and use artifacts to realize their functional goals, but they do not apply to the agents/users themselves. Thus, while the notion of computational artifact functions adopted here specifies roles for agents that use artifacts for computation, it specifies nothing about the origins of agent capacities, functions, goals, or actions—or whether there is any sense in which these origins are, at bottom, themselves computational. The present theory thus has nothing to say about computational theories of cognition (e.g. Piccinini 2016), except that any notion of computational function invoked in such theories must necessarily be of a different nature than are the computational functions of artifacts (e.g. computing machines) adopted here. Instrumental computation—computation achieved by agents through artifact use—thus differs distinctly both from computation achieved by agents via their own computational capacities and from would-be intrinsic computation defined exclusively in terms of physical transformations of inanimate objects. It accommodates the former and rejects the latter.
- While the present paper focuses on digital computation for concreteness, and while the IPA theory is consequently elaborated specifically for DCAs, the general ideas about computing artifacts and instrumental computation presented here extend to non-digital forms of computation. Computation—digital or otherwise—is a subclass of information processing, so digital and non-digital computing artifacts belong to different subclasses of information processing artifacts as they are defined and characterized by the present theory. This theory could thus be elaborated for non-digital computing artifacts just as it is elaborated for DCAs in this work, given a sufficiently precise characterization of the relevant non-digital form of computation.

[3] For a discussion of instrumental artifact functions in the context of other philosophical notions of function, see Sect. 1 of Hughes (2009).

The remainder of this paper is organized as follows. In Sect. 2, I sketch essentials of Ladyman's ideal classical *L*-machines and Hughes' theory of instrumental artifact functions. Next, in Sect. 3, I provide an instrumental function ascription for ideal digital computing artifacts—artifacts based on physical realizations of ideal *L*-machines—that can be used by agents to evaluate logical functions, and I identify the essential ingredients of these ideal *L*-IPAs (3.1–3.4). I then discuss distinctions I recognize between physical evolution, computation and information processing, and how they enable distinction between *L*-IPAs and systems with identical computational capacities that are not computing artifacts (3.5). In Sect. 4, I then consider the ascription of deterministic computational functions to non-ideal computing artifacts, and discuss the implications of artifact non-idealities for the association of physical artifact capacities with unique logical functions. I briefly review Hughes' notions of artifact failure, malfunction, and normality (Sect. 4.1), as well as Anderson's generalized (non-ideal) *L*-machines and associated efficacy measures (Anderson 2010). With this, I formally define non-ideal computing artifacts and associated reliability and effectiveness measures (Sect. 4.2). This definition clarifies the sense in which a real, imperfect artifact can unambiguously be ascribed the function of executing a particular computation (a) without possessing the capacity to reliably and effectively do so or (b) by possessing this capacity only in a trivial sense. Finally, I revisit the key distinction between function and capacity in instrumental computation, clarify its role in distinguishing computers from other physical systems, and discuss its implications for intrinsic, unconventional, and natural computation (Sect. 5). The paper concludes in Sect. 6.

2 Preliminaries

For concreteness, I focus here on physical systems that can be used by agents to evaluate logical functions *L*—hereafter "logical transformations". This is a restricted subclass of IPAs, and even of DCAs, but it is broad enough to include any artificial or natural system that can be used to evaluate a logical transformation *L* of arbitrary complexity for any input—suitably encoded in the artifact's initial physical state[4]—drawn from the domain of *L*. Formal characterization of any such artifact requires specification of what—beyond a physical system's capacity for use in implementation of a logical transformation *L*—is required for a system to be ascribed the function of implementing *L*, including what an agent's use of the system for this purpose entails.

[4] This requirement would seem to exclude as *L*-IPAs those systems that respond to new external inputs *during* execution, such as finite-state automata (FSA) driven by external inputs, von Neumann processors that fetch instructions from and exchange data with an external memory, and Turing machines that interact with an external tape. However, such systems *can* be described as *L*-IPAs if the external sources and targets—the buffer holding an FSA input string, the von Neumann processor's instruction and data memory, the Turing machine's tape—is finite and is internalized as part of the artifact.

2.1 *L*-Machines and Computational Capacity

The capacity of a physical system to implement an abstract logical transformation L is captured, at the right level of generality in idealized situations, by the concept of an ideal L-machine. An abstract M-input, N-output logical transformation L is a mapping

$$L : \{x\}_L \to \{y\}_L$$

of M discrete logical inputs $x_i \in \{x\}_L$ into N discrete logical outputs $y_j \in \{y\}_L$ in accordance with a rule

$$L(x_i) = y_j \quad \forall i \in \{i\}_j \tag{1}$$

where $\{i\}_j = \{i|L(x_i) = y_j\}$. For example, the Boolean NOR function is a logical transformation that maps $M = 4$ inputs $\{x\}_{NOR} = \{00, 01, 10, 11\}$ into $N = 2$ outputs $\{y\}_{NOR} = \{0, 1\}$ according to the rule $L(01) = L(10) = L(11) = 0$ and $L(00) = 1$ (i.e. $\{i\}_0 = \{01, 10, 11\}$ and $\{i\}_1 = \{00\}$).

A physical device $\mathcal{D}$ has the capacity for implementation of L if its dynamics support a mapping

$$\Lambda_L : \{D^{(in)}\} \to \{D^{(out)}\}$$

from distinguishable physical input states $D_i^{(in)} \in \{D^{(in)}\}$ of $\mathcal{D}$ into distinguishable physical output states[5] $D_j^{(out)} \in \{D^{(out)}\}$ of $\mathcal{D}$ such that

$$\Lambda_L(D_i^{(in)}) = D_j^{(out)} \quad \forall i \in \{i\}_j. \tag{2}$$

Here Λ_L is an evolution operator describing the initial-to-final physical state transformations of $\mathcal{D}$ generated by its dynamics. An L-machine, as defined by Ladyman (Ladyman et al. 2007), is the four-tuple $\{\mathcal{D}, \{D^{(in)}\}, \{D^{(out)}\}, \Lambda_L\}$. I will call this construction an *ideal* L-machine to distinguish it from its generalization by (Anderson 2010) that also plays a significant role in this work. Note that specification of an M-input, N-output L-machine implementing an M-to-N logical transformation L requires specification of M distinct initial-to-final physical state transformations $\Lambda_L(D_i^{(in)})$.

Several aspects of ideal L-machines warrant emphasis at this stage. First, ideal L-machines are noiseless, which is to say that the physical output states $D_j^{(out)}$ corresponding to the various logical outputs y_j can be perfectly distinguished from one another. This ensures that, by performing an appropriate measurement on $\mathcal{D}$, a user can unambiguously "read" the logical output from the evolved state of $\mathcal{D}$. Noisy L-machines are non-ideal by definition: they are necessarily generalized L-machines.

[5] The physical states $D_i^{(in)}$ and $D_j^{(out)}$ are generally statistical states.

Second, ideal L-machines that implement surjective L—logical transformations L for which $M > N$[6]—do so faithfully. This is to say that for all logical inputs x_i that map into the same logical output y_j, the physical input states representing these x_i (the $D_i^{(in)} \in \{D_i^{(in)} | i \in \{i\}_j\}$) evolve into the same physical output state via Λ_L (i.e. $\Lambda_L(D_i^{(in)}) = D_j^{(out)} \quad \forall \{D_i^{(in)} | i \in \{i\}_j\}$). Faithful physical output states are purged of any structure that would reveal more about their "input of origin" than could be inferred from the logical function L being implemented: the physical state mappings mirror the "many-to-oneness" of L. Unfaithful L-machines are also non-ideal by definition: they are necessarily generalized L-machines as well.

Third and finally, while the device $\mathcal{D}$ of an ideal L-machine clearly has the *capacity* for implementation of L, the question of what it means to *have the function* of implementing L—or what all is involved in its *use* for this purpose—is not addressed in the definition of an L-machine. Computational capacities are obviously necessary for successful use of a system for computation, but they are insufficient. Here I will follow Anderson (2017) and refer to physical systems that possess computational capacities as *protocomputing* systems to clearly distinguish them from what we colloquially call "computers"—artifacts that not only possess such capacities, but that can also be regarded as having computational functions and used to execute computations. Characterization of this distinction requires a theory of artifacts that appropriately addresses function ascription and use.

2.2 Artifacts: Function and Use

I address questions of artifact function and use through the artifact theory of Hughes (2009), augmenting and supplementing his theory as necessary for extension to IPAs and DCAs. Hughes' theory is particularly well suited to this purpose[7] for two reasons. First, there is a natural fit between Hughes' notion of artifact malfunction and the computational efficacy measures formulated for generalized L-machines (Anderson 2010), facilitating the formal characterization of non-ideal L-IPAs. Second and more generally, Hughes' emphasis on instrumental user knowledge over explanatory knowledge seems appropriate in computational contexts. Real-world computation, as noted earlier, overwhelmingly involves users who are adept at using their devices for numerous tasks but have little or no explanatory knowledge of what gives rise to the devices' enabling computational capacities. Many proficient users will not even recognize some of the tasks that they perform with digital devices as being of a computational nature.

According to Hughes' artifact theory, an instrumental function ascription comprises four essential elements (Hughes 2009):

[6] Note that while the definition of an ideal L-machine accommodates machines that implement logical transformations for which $M = N$ (i.e. bijective, invertible, or logically reversible L), faithfulness is meaningful only for logical transformations for which $M > N$ (surjective, noninvertible, or logically irreversible L).

[7] This is not, however, to say that other artifact theories (e.g. Houkes and Vermaas 2010) could not be similarly applied to computing artifacts.

1. *Functional Goal* φ: The aim of the function, i.e. "a condition that can be realized by proper use" of the artifact. While the condition may be achieved with varying degrees of success, there is a fact of the matter of whether or not the goal has been achieved on any given use.
2. *Use Plan* α: A prescription for how one should manipulate the artifact and related objects in the context of use in order to realize the goal. The use plan includes specification of "allowable inputs."
3. *Contexts of Use* C: Specification of normal contexts of use, including a set C of situations $c \in C$ for which use of the artifact is likely to result in a suitable outcome.
4. *Artifactual Type* T: The class of artifacts to which the instrumental function applies. Types are defined at a sufficiently narrow level that all tokens t of that type share a common use plan. Functions are ascribed to artifacts of type T, not to individual tokens $t \in T$.

Hughes denotes such a function ascription as $\langle \varphi, \alpha, C, T \rangle$, and, in his initial sketch, states the following necessary and jointly sufficient conditions for an instrumental function ascription to be true (Hughes 2009):

IF(a) In situations satisfying C, using a T-token as prescribed by α is a means to φ.
IF(b) Some causally-relevant persons value this capacity of T-tokens as above.

IF(a) specifies when and how tokens of a given artifactual type function as means to the end φ. **IF**(b) ensures that would-be function ascribers are causally relevant to the physical systems upon which they confer functions. The causally-relevant persons of **IF**(b) recognize that there are actions one could take to realize φ—to achieve this functional goal—through use of the artifact, not just passive observers who imagine how it might happen to come about or have happened to come about.

Hughes' theory, adapted to computing artifacts, has the potential to capture both appropriately liberal and appropriately conservative aspects of computational function ascription. It would accommodate the ascription of computational functions to both conventional and radically unconventional physical substrates, with successful use of artifacts based on these substrates realistically constrained by their intrinsic capacities to serve as means to the computational ends of the agents who would use them. Additionally, a Hughes-like theory of IPAs would unambiguously reject systems like rocks, walls, and pails of water—systems to which pancomputationalists have attributed computational capacities—as computing artifacts (read: "computers"), and would do so on clear and principled grounds that supplement the usual objections to triviality arguments about computation.[8] Whatever the alleged

[8] The usual triviality arguments about computation are really triviality arguments about computational *capacity*, which presumes that display of computational capacity amounts to implementation of computation. By contrast, instrumental computation regards both intrinsic computational capacities—objectively grounded in artifact structure and composition—*and* agent-ascribed computational functions as necessary components of implementation. In this work I argue separately against both the trivialization of computational capacities (Sect. 4.2) and the possession of intrinsic computational functions by physical objects (Sect. 5).

capacities of these systems, a Hughes-like IPA theory would deny them computational functions because they lack plausible use plans and would-be users who regard these systems as potential means to their computational ends.

I show below how Hughes' theory can be retrofitted to accommodate essential features of IPAs in general and DCAs in particular. The resulting theory, constructed specifically for DCAs based on ideal L-machines, addresses the roles that allowable inputs play in the specification of computational function and that auxiliary systems play in the use of physical systems for computation. It also addresses implications of the abstract nature of computational functions—and the multiple realizability of their physical implementations—for the characterization of DCAs and their use. This will lay most of the groundwork for characterization of more general and more realistic DCAs in Sect. 4.

3 Ideal Computing Artifacts

I now consider the device $\mathcal{D}$ of an L-machine, regard it as an artifact that is to be ascribed the function of implementing the logical transformation $L(x)$, revisit the essential elements of function ascription from Hughes' theory in this context, and incorporate required modifications into a Hughes-like L-IPA function ascription.

3.1 Functional Goal

The functional goal of an L-IPA can be stated simply as **Evaluate** $L(x)$, where x is a user-selected input drawn from the set $\{x\}_L$. This has superficial similarities to the functional goals of ordinary artifacts such as staplers, corkscrews, and hammers, but differs in two essential respects that are crucial for what follows. To highlight these differences, I contrast the functional goal **Evaluate** $L(x)$ of an L-IPA with the functional goal **Fasten**(x) of a stapler used as an example in Hughes (2009).

Both **Evaluate** $L(x)$ and **Fasten**(x) accommodate multiple inputs, with x selected from the finite set $\{x\}_L$ of logical inputs to $L(x)$ in the former case and from appropriate stacks of paper in the latter. Yet the nature of the allowable inputs differs fundamentally for these two functional goals, as do their relationships to the functional goals and the physical objects used by agents to realize these goals.

The first and most obvious difference is the categorical difference between the nature of the allowable inputs associated with L-IPAs and with staplers. The allowable inputs x to **Evaluate** $L(x)$ are arguments of an abstract logical function, whereas the allowable inputs x to **Fasten**(x) are concrete stacks of paper. A consequence of this for L-IPAs is a type mismatch between the inputs associated with the functional goal (abstract logical inputs $x \in \{x\}_L$) and the corresponding inputs to the device $\mathcal{D}$ that an agent uses as a means to realize this goal (physical states $D_i^{(in)}$ of $\mathcal{D}$). This necessitates specification of a what is typically called a representation relation (or encoding) $\{x_i \rightarrow D_i^{(in)}\}$ for an L-IPA, which must be reflected in its use plan (see Sect. 3.2). There is no such type mismatch for **Fasten**(x) and thus no need for such a representation relation. The inputs referred to in the functional goal of a stapler and the inputs to a physical stapler that will be used to realize this goal are of the same type—they are concrete stacks of paper.

The second (and perhaps less obvious) difference between the functional goal **Evaluate** $L(x)$ and a functional goal like **Fasten**(x) is the difference in the roles that allowable inputs play in defining the artifact function. The functional goal **Evaluate** $L(x)$, defined in terms of *a mapping* $L(x)$ from the full set $\{x\}_L$ of logical inputs to the full set $\{y\}_L$ of logical outputs, is input selective: various x drawn from a given subset of $\{x\}_L$ may produce the same output, whereas various x drawn from multiple subsets of $\{x\}_L$ generally yield *different* $L(x)$. Such is not the case for a functional goal like **Fasten**(x): the goal of "fastening" is defined without reference to various subsets of stacks of paper and the different ways that they must be acted upon to achieve the goal of fastening. **Fasten**(x) is achieved by producing the same result for all allowable input stacks of paper.[9]

It follows from the above that an input-selective functional goal like **Evaluate** $L(x)$ cannot be achieved in a single use of an L-IPA. Because the functional goal is defined in terms of a set of inputs, an artifact that (by some representation relation) happened to map a single input x_i into an output $y_j = L(x_i)$ on a particular use could not establish that artifact as a means to the end **Evaluate** $L(x)$. Nor could it establish as much on multiple uses involving inputs selected from any (proper) subset of the allowable inputs $\{x\}_L$—multiple uses involving *all* possible inputs would be required.[10] Clearly, this is not the case for an input-insensitive functional goal like **Fasten**(x). Even the single use of a stapler that fastens the papers in one particular stack demonstrates that the stapler can be used as a means to the end of fastening, and a (properly functioning) stapler will respond in precisely the same way to all stacks of paper that it can handle. A stapler can display its capacity to for fastening in a single use on one stack of paper.

The implication of this second difference for an L-IPA is that any appeal to its function as an implementor of the multiple-input logical function $L(x)$—or any claim regarding the efficacy with which it implements L or the resources this requires—is necessarily a statement about the artifact's response to *all* of its allowable (logical) inputs. It is a corollary that its capacity to implement the function **Evaluate** $L(x)$ can only be displayed over multiple uses that sample the full set $\{x\}_L$ of allowable inputs.

I accommodate this feature of an L-IPA functional goal φ_L simply by restating this goal as "**Evaluate** $L(x)$ for arbitrary, user-selected input $x_i \in \{x\}_L$", provided that (*i*) φ_L is understood as a set $\varphi_L = \{\varphi_i\}_L$ of *constitutive goals* φ_i, where φ_i is the input-specific mapping $x_i \rightarrow L(x_i)$, and (*ii*) specification of an appropriate, realization-specific input representation relation $\{x_i \rightarrow D_i^{(in)}\}$ is reflected in the use plan. It is the constitutive goals φ_i that can, as a matter of fact, be met or fail to be met on

9 While it is certainly the case that the limited capacities of a given type of stapler constrain the allowable inputs to the functional goal **Fasten**(x)—e.g. the thicknesses of stacks of paper than can be fastened by a particular model of stapler produced by a particular manufacturer—these constraints are not defining characteristics of the functional goal. They are parameters in the contexts of use for the stapler.

10 Analogously, the capacity of a mechanism for sorting coins could not be revealed by feeding it one or more pennies and seeing what happens, simply because the functional goal of sorting coins is defined in terms its response to *all* of its allowable inputs (e.g. {penny, nickel, dime, quarter}).

any individual use, whereas achievement the functional goal φ_L = **Evaluate** $L(x)$—dependent as it is on different artifact responses to different inputs—can not.

3.2 Use Plan

An L-IPA's capacity to implement $L(x)$ is objectively rooted in the physical dynamics that evolve initial states $D_i^{(in)}$ of device $\mathcal{D}$ into final states $D_j^{(out)}$. Yet, an agent's ability to make use of this capacity—and thus for an L-IPA to fulfill the functional goal **Evaluate** $L(x)$—requires more. Specifically, it requires additional capacities to prepare the initial state $D_i^{(in)} \in \{D_i^{(in)}\}$ corresponding to any logical input x_i selected by the user at will; to activate, drive, or enable the dynamical evolution of $\mathcal{D}$ via Λ_L so its states are transformed in the desired manner (*cf.* Sect. 2.1); and to unambiguously read the resulting $D_j^{(out)}$ corresponding to the logical output $y_j = L(x_i)$. Furthermore, it requires elaboration of these operations in the form of an L-IPA use plan that would enable an agent to use the device $\mathcal{D}$ of an ideal L-machine to evaluate $L(x)$ for any input $x \in \{x\}_L$ of their choosing. Additional physical systems required for these operations—the "related objects" of Hughes' use plan characterization—must also be introduced. These auxiliary systems must include components that are directly accessible to the user, since, without an appropriate physical interface, users will generally not be able to access, manipulate, and observe the device $\mathcal{D}$ in a manner that would enable them to use it to evaluate $L(x)$.

The essential ingredients of an L-IPA use plan are thus **Load**(x), **Apply**(Λ_L), and **Read**(y) operations. **Load**(x) and **Read**(y) are user-initiated physical operations that correlate the physical state of the device $\mathcal{D}$ to states of physical systems that are external to $\mathcal{D}$ but that (i) are also directly accessible to the user and (ii) that physically instantiate logical inputs and outputs by specified representation relations. **Load**(x) brings the state of $\mathcal{D}$ into correlation with that of an external input referent system $\mathcal{R}_{in}$, which physically instantiates the logical input x_i in referent state r_i of $\mathcal{R}_{in}$. ($\mathcal{R}_{in}$ is essentially part of a physical "input file", e.g. a particular row and column of a ledger or a location in an input buffer or a computer memory holding input data.) **Read**(y) brings the state of an external apparatus $\mathcal{M}$ into correlation with the state of $\mathcal{D}$, where $\mathcal{M}$ physically instantiates the measurement outcome y_j in apparatus state m_j (e.g. an alphanumeric display, computer screen, or location in an output buffer or computer memory storing output data). **Apply**(Λ_L) is the intermediate step in which input states are evolved into output states via the evolution operator Λ_L. This step may require that the user take action, or, in some cases, that they simply let a specified amount of time pass between **Load**(x) and **Read**(y).

The use plan for the elementary L-IPAs considered here can thus be expressed simply as the ordered set α_L = (**Load**(x), **Apply**(Λ_L), **Read**(y)), where the input states of $\mathcal{D}$ are evolved to the output states in the **Apply**(Λ_L) step. The **Load**(x) and **Read**(y) operations and associated physical systems provide the interface required for use, differentiating a system with physical capacities for implementation of $L(x)$ from an device that could be used as an L-IPA for this purpose. After each use of an L-IPA implementing a logical transformation $L(x)$, the joint state of $\mathcal{R}_{in}\mathcal{M}$ physically instantiates the input-output pair $(x_i, L(x_i) = y_j)$ relevant to that use and accessible the user. Note that the preparation and measurement processes required for artifact use incur their own physical costs—some unavoidable even in principle—giving physical teeth to the distinction between display of computational capacity on the one hand and use for computation on the other.

3.3 Contexts of Use and Artifactual Types

In Hughes' theory, artifact types are associated with function ascriptions. For an L-IPA, an artifactual type T_L would thus be associated with a type of device $\mathcal{D}$ that, when used according to α_L in contexts C_L, is a means to evaluation of $L(x)$. Additionally, for reasons discussed by Hughes, it is important that artifactual types are defined at a "suitably narrow" level. He takes this to be a level sufficiently narrow that "a single user plan suffices for every token of that type"—at the level of the artifact design (Hughes 2009). For an L-IPA, this would suggest specification of types roughly as narrow as "Consolidated Semiconductor Corporation's standard-cell two-input NOR gate realized in their 22 nm silicon CMOS process technology." For every type defined at this level of narrowness, there will be very specific contexts of use C_L specifying the environmental and operating conditions (e.g. temperature, power supply) that tokens of that particular type require to play their intended roles in ideal L-machines.

Because computational implementations are multiply realizable, however, these two features of Hughes' theory are in conflict for L-IPAs. There may be many kinds of devices of radically different design, configuration, and composition that can all serve as means to the same computational end of evaluating $L(x)$. But L-IPAs based on each kind of devices will have its own L-machine specification ($\{D_i^{(in)}\}$, $\{D_j^{(out)}\}$ and Λ_L), use plan, and contexts of use, and these may differ wildly for different kinds of devices. There are, for example, radical differences between what it takes to implement the logical NOR function with a silicon electronic NOR gate and what it takes to implement NOR with, say, DNA (Okamoto et al. 2004) or slime mold (Adamatzky 2015). Yet, all three realize the same functional goal of implementing NOR, and all three could even draw their inputs from the same input referent $\mathcal{R}_{in}$ and register their outputs in the same measurement register $\mathcal{M}$ without an observer (of $\mathcal{R}_{in}\mathcal{M}$) being to able to tell which particular NOR gate realization is "under the hood".

Since the defining functional goal of an L-IPA can be realized by multiple means—multiple kinds of devices each with their own idiosyncratic use plans—it cannot be the case that a single use plan suffices for every L-IPA token. Thus, an L-IPA function ascription cannot be both common to all artifacts with the goal **Evaluate** $L(x)$ and at the same time sufficiently narrow to specify how the artifact is to be used to achieve this goal. A Hughes-like L-IPA function ascription can apply to a very specific type of L-IPA that implements $L(x)$ in its own specific way, but not to all L-IPAs that can realize the same functional goal.

This is easily remedied by defining L-IPA artifactual types at two levels. At the higher (user) level, I associate the broad artifactual type T_L with the functional goal **Evaluate** $L(x)$ so it includes all devices that can be used to realize this goal (if in various device-specific ways). At the lower (device) level, I associate a narrower artifactual subtype $T_L^{(k)}$ with artifacts of type T_L that are based on devices $\mathcal{D}^{(k)}$ of a particular design, structure, and composition. Every token of artifactual type T_L belongs to some subtype $T_L^{(k)} \in T_L$. Every token of artifactual subtype $T_L^{(k)}$ is based on a device $\mathcal{D}^{(k)}$ that can be used—via application of appropriate use plan $\alpha_L^{(k)}$ in appropriate contexts $C_L^{(k)}$—to achieve the goal **Evaluate** $L(x)$. Thus, although an L-IPA type T_L is defined at the higher level of functional goal, use plans and conditions of use for specific realizations—and therefore instrumental function ascriptions—are defined at the lower, realization-specific level.

Through this association of artifact types with functional goals and artifact subtypes with use plans, Hughes' theory is thus extended to accommodate the multiple realizability of computational implementation. With this and the other considerations discussed above, I now specify Hughes-like function ascriptions for L-IPAs.

3.4 Function Ascription for Ideal Computing Artifacts

I denote the instrumental function ascription for an ideal L-IPA of type T_L and subtype $T_L^{(k)}$ as

$$\langle \varphi_L, \alpha_L^{(k)}, C_L^{(k)}, T_L^{(k)} \rangle_L$$

with the elements

1. *Functional Goal* $\varphi_L = \{\varphi_i\}_L$: **Evaluate** $L(x)$ for arbitrary, user-selected input $x_i \in \{x\}_L$, where φ_i is the constitutive goal $x_i \rightarrow L(x_i)$.
2. *Use Plan* $\alpha_L^{(k)}$: (**Load**$^{(k)}(x)$; **Apply** $\Lambda_L^{(k)}$; **Read**$^{(k)}(y)$), where **Load**$^{(k)}(x)$ is a state-preparation procedure that conditionally prepares device $\mathcal{D}^{(k)}$ in state $D_i^{(in)}$ when the user-selected logical input is x_i, $\Lambda_L^{(k)}$ is an evolution operator that governs transformation of the physical states of device $\mathcal{D}^{(k)}$, and **Read**$^{(k)}(y)$ is a specified measurement process—a measurement performed on device $\mathcal{D}^{(k)}$—that always yields a discrete outcome m_j associated with one of the possible logical outputs y_j.
3. *Contexts of Use* $C_L^{(k)}$: Specification of normal contexts of use, including a set $C_L^{(k)}$ of situations $c \in C_L^{(k)}$ for which use of an artifact of subtype $T_L^{(k)} \in T_L$ according to $\alpha_L^{(k)}$ is likely to result in a suitable outcome.
4. *Artifactual Subtypes* $T_L^{(k)}$: Artifacts of type T_L based on devices $\mathcal{D}^{(k)}$ of subtype k, defined by their particular design, configuration, and/or composition.

Note that two conditions must be met for an ideal L-IPA is to serve as a reliable means to the end of evaluation $L(x)$. First, the readout measurement must be properly selected. Specifically, on every use of the L-IPA that produces the output state $D_j^{(out)}$ of device $\mathcal{D}^{(k)}$, the outcome m_j of the **Read**$^{(k)}(y)$ operation corresponding to the logical output y_j must always registered. The output states of ideal L-machines are mutually distinguishable by definition, so such readout measurements necessarily exist. Second, the operating conditions must conform to those specified in the contexts of use $C_L^{(k)}$, so the device $\mathcal{D}^{(k)}$ transforms states precisely in the manner required by the ideal L-machine definition.

This theory of ideal L-IPAs comports with the commonsense view of what distinguishes computers from other physical systems—that they are used by agents to perform particular computations—while explicitly addressing less obvious challenges faced in supporting this view. Specifically, L-IPAs are distinct from other physical systems in the extrinsic sense that they are used by agents for computation, with nothing about the dynamics of the device taken to be intrinsically computational. L-IPAs are computational precisely because they are ascribed computational functions by agents and are used by agents to realize their computational goals—the evaluation of particular logical functions.

Two significant challenges remain. First, although physical artifact dynamics are not taken to be intrinsically computational in the ideal L-IPA theory, ideal L-IPAs are defined so they include only those systems with dynamics that are ideally suited for use in implementing L. The device $\mathcal{D}^{(k)}$ of an ideal L-IPA *does* evolve the physical input states in a manner that objectively mirrors the computational state mapping associated with a particular computation—the logical function L. This rules out applicability of the ideal L-IPA theory to most real L-IPAs. As noted in Sect. 1, physical state transformations in real computing artifacts may mirror computational state mappings quite imperfectly, sometimes even by design as a concession to energy efficiency, circuit complexity, and/or other considerations. In Sect. 4, I show how the dynamical restrictions that define ideal L-IPAs can be relaxed—with the consequences of this relaxation for computational success rigorously quantified—generalizing L-IPAs without introducing ambiguity into the notion of computational artifact function.

Second, because L-IPAs are distinguished from other physical systems *only* in the extrinsic sense, they are computers only because they have agent-ascribed computational functions. They are denied computational functions even if they possess the intrinsic capacities required for their use in executing computations. By the same token, L-IPAs perform computations only when they are being used for this purpose. L-IPAs not in use by agents for computation cannot be considered to execute computations even when they evolve exactly as they do when they are so used. This obviously counterintuitive aspect of instrumental computation requires justification, and will be discussed further in Sect. 5.

3.5 Dynamics, Computation, and Information Processing

I close this section by highlighting several key distinctions that locate ideal digital computing artifacts (like L-IPAs) in the broader context of evolving physical systems. This will serve to clarify the relationships between the ideas presented so far. I also consider how these relationships generalize to—and are modified for— non-digital computing artifacts, as well as for the non-ideal digital computing to be artifacts discussed in Sect. 4. Figure 1 provides a visual guide.

The first of these key distinctions is the distinction between physical systems that possess computational capacities and those that do not. This distinction relies upon a physical characterization of computational capacity like that provided by the definition of an ideal L-machine (Sect. 2.1). I define *Protocomputing Systems* as physical systems that possess such computational capacities. They may or may not also be artifacts and may or may not process information, as indicated in Fig. 1. What counts as a protocomputing system is, of course, relative to a particular type of computation (e.g. digital computation).

The second key distinction is the distinction between physical systems that are used by agents to achieve functional goals and those that are not. This distinction is made in theories of artifacts, as they are in Hughes' theory (Sect. 2.2) and its extension to computing artifacts provided by this work. Such systems are denoted simply as *Artifacts* in Fig. 1. Note that not all artifacts are taken to have capacities for information processing or for computation, as per common sense but contrary to the claims of pancomputationalism.

The third key distinction, not yet discussed in this work, is the distinction between physical systems that process information and those that do not. This distinction clearly relies upon a definition of information processing, and, for present purposes, one that is distinct from computation. Here I adopt the physical conception of information elaborated and defended in Anderson (2017). Three essential features of this conception of information are relevant to the present work: information is physical, relational, and observer relative. Specifically, the amount $I^{\mathcal{RD}}$ of information that an observer-accessible physical system $\mathcal{D}$ holds *about* another observer-accessible physical system $\mathcal{R}$ is associated with physical correlations between the states of $\mathcal{R}$ and $\mathcal{D}$, and is quantified by a suitable measure defined on the joint physical state of $\mathcal{RD}$.

On this physical conception of information, any dynamical evolution of a system $\mathcal{D}$ that holds $\mathcal{R}$-information ($I^{\mathcal{RD}} > 0$) processes $\mathcal{R}$ information.[11] All evolving information-bearing systems are thus *Information Processing Systems*. As indicated in Fig. 1, information processing systems need not have computational capacities—either used or unused—nor do they need to be used or usable by any agent for information processing. However, because physical correlations are necessarily created and/or destroyed in use of an artifact for computation, use for computation necessarily entails information processing while display of computational capacity (protocomputing) does not.

With this, we can more clearly define information processing artifacts and computing artifacts and see how they are related. *Information Processing Artifacts* are physical systems that process information and are used by agents to achieve information processing ends. *Computing Artifacts* (including but not limited to DCAs) are those information processing artifacts that possess computational capacities and are used by agents as means to their computational ends. All computing artifacts are IPAs: the use of a physical system (e.g. the device $\mathcal{D}$ of an L-machine)

[11] This includes, as trivial special cases, evolutions that simply preserve information. While it may seem counterintuitive to regard the preservation of information as even a trivial form of information processing, there are many familiar information processing systems whose function requires that they preserve information. These include noiseless communication links, reversible computers, and memory systems.

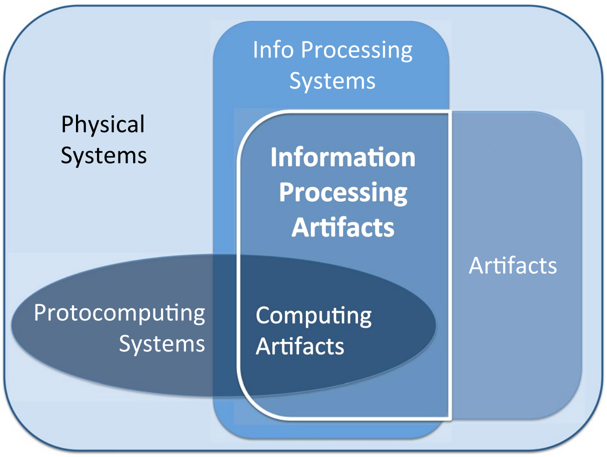

Fig. 1 Visual representation of key definitions used in this work and the relationships between them. Computing Artifacts—colloquially "computers"—are physical systems with computational capacities (read: are Protocomputing Systems) that are used by agents (read: are Artifacts) as means to their computational ends. They are a subclass of Information Processing Artifacts, since their use for computation entails processing of information about inputs that are instantiated in an external source

for computation requires establishment of correlations between the physical state of the device and an external instantiation of the computational input (e.g. in the input referent $\mathcal{R}_{in}$) as discussed in Sect. 3.2. The converse is, however, not true: artifacts obviously need not possess or display computational capacities for them to be used by agents to implement information processing tasks that are not of a computational nature.[12]

Note that, on this characterization of digital computing artifacts, computational functions have a semantic dimension while computational capacities do not. The possession and display of an artifact's computational capacities are rooted solely in its structure, composition, and state-transformation characteristics. However, use of an artifact by an agent as a means to computational ends has an inescapably semantic dimension: computing artifacts process (non-natural) semantic information[13]

[12] A garden-variety audio amplifier under normal use is, for example, an IPA—it unambiguously processes information about an input source—but it is not a DCA since it does not perform digital computations.

[13] Non-natural and natural semantic information are discussed and distinguished in Piccinini and Scarantino (2011) and Piccinini (2015). Note that computing artifacts necessarily process non-natural semantic information when they are in use, but the same is not true of evolving protocompting systems *even when they are displaying their computational capacities*. Evolving protocomputing systems that are not computing artifacts may bear and process *natural* semantic information—bare physical correlations between their states and the states of external systems—or they may bear and process no information at all. This, among other things, distinguishes the present work from the abstraction/representation theory of Horsman et al. (2018), which requires representation for computation but only in the sense of natural semantic information. Their "representational entities" are, like the referent systems of this work, necessarily physical, but in their theory the physical processing of natural representations counts as computation with no role for agents.

when they are so used. Use plans for computing artifacts necessarily include the representation relations that would imbue physical states with semantic content, enabling abstract logical functions to be evaluated through the use of concrete physical objects. Agents imbue artifact states with semantic content just as they imbue artifacts with computational functions.[14]

Finally, I consider adjustments to the framework of Fig. 1 that would be required for two other classes of computing artifacts. First, consider computing artifacts that are used to perform non-digital computations of some specified type. The subset of evolving physical systems that have the requisite capacities would differ from the subset with capacities for digital computation, so the *Protocomputing Systems* region of Fig. 1 would differ for this type of non-digital protocomputing systems and digital computing systems. The *intersections* of that region with the regions representing *Info Processing Systems* and *Information Processing Artifacts* would consequently differ, but these two regions themselves would not—they include all systems and artifacts that process and are used to process physical information as defined above. Second, consider DCAs that do not have the capacity to reliably transform states as required for execution of a particular digital computation, but that are unambiguously used by agents for execution of that computation with some degree of success. All real digital computing devices are strictly speaking of this nature, as noted earlier. This would severely blur the boundaries in Fig. 1 between digital *Protocomputing Systems* and all other *Physical Systems*, which are sharp only for ideal DCAs, but again would leave the *Info Processing Systems* and *Information Processing Artifacts* regions unchanged—they are not defined with reference to any particular form of computation. The boundaries of the *Information Processing Artifacts* and digital *Computing Artifacts* regions would remain sharp, as artifact functions are agent ascribed and are unambiguous even when capacities are not. The *Computing Artifacts* region would also be larger for non-ideal DCAs than it would be for ideal DCAs, as it would include all real, imperfect artifacts used for digital computation.

This second case is relevant to the next section, where I formally characterize non-ideal *L*-IPAs. Computational functions are ascribed to non-ideal *L*-IPAs as unambiguously as they are ascribed to their ideal counterparts, but shades of grey in computational efficacy are recognized, accommodated, and rigorously quantified in the theory of non-ideal *L*-IPAs.

[14] The central role of agents in defining artifact function, and the distinction between artifact function and capacity, is perhaps easiest to see in the case of natural systems appropriated by agents to achieve their goals. Agents use fallen trees to cross rivers and use canaries to detect lethal gasses in coal mines, having recognized that trees and canaries intrinsically possess the required capacities. Yet, we would not say that trees inherently have the function of bridging rivers, or that canaries have the function of detecting lethal gasses, independent of agents that appropriate and use them for these purposes.

4 Non-ideal Computing Artifacts

Users of ideal L-IPAs realize their constituent functional goals φ_i on every use. **Load**(x) always prepares the device in the state $D_i^{(in)} \in \{D^{(in)}\}$ corresponding to the user-selected input x_i, **Apply** (Λ_L) always evolves this state into the output state $D_j^{(out)} \in \{D^{(out)}\}$ corresponding to the logical output $y_j = L(x_i)$, and **Read**(y) always correctly registers the outcome m_j in the measurement apparatus $\mathcal{M}$. The ideal L-machine on which an ideal L-IPA is based is consistent with realism about computation in the qualified sense of Ladyman (2009): "it is an objective fact that it implements a particular transformation structure since its ability to do so depends on the structure of the system and the laws governing its time evolution and is independent of exactly what the physical states of the system are taken to represent." The transformation structure of an ideal L-machine uniquely and unambiguously mirrors that of the logical transformation L that it is used to implement.

However, real information-processing artifacts—subject as they are to environmental noise, design flaws, token-to-token variations, non-optimal use plans, and/or breakdown of expected behavior at the nanoscale—cannot live up to this standard; they realize their functional goals only imperfectly. While this reduced functional efficacy is undesirable, and great effort is typically expended to minimize it in manufactured computing artifacts, it is unavoidable. It is even tolerated or embraced in some emerging computing paradigms—e.g. approximate, noise-tolerant, nanoscale, quantum, and probabilistic computing paradigms—as a concession to other considerations such as design simplicity, energy efficiency, artifact-application compatibility, feasibility, and cost. Whatever the nature or extent of their imperfections, no real digital computing artifacts are ideal in the sense of ideal L-machines.

Ideal L-machines are thus an inadequate basis for realistic L-IPAs, in which physical state transformations *do not* objectively and uniquely mirror the abstract logical transformation the artifact is used to implement. This motivates relaxation of the conditions on physical state transformations in the definition of an L-machine, appropriately generalizing this definition so it accommodates the imperfections of real devices and serves as a basis for characterizing non-ideal IPAs. Relaxation of these conditions, realistic as it is, forfeits the unambiguous association of the state transformation structure of physical devices with specified logical transformations L. This opens the question of what distinguishes "computers" from other physical systems if particular computations cannot be uniquely associated with a system's physical capacities. Does it render computational capacities subjective or in some sense trivial, opening a door to pancomputationalism? Or does it imply that no real physical artifact can have the function of evaluating a particular L? If neither, then how can an artifact unambiguously have the function of evaluating a particular L without having the unambiguous capacity to do so?

Hence the challenge for a theory of non-ideal L-IPAs. If such a theory is to distinguish real computing artifacts that have the function of evaluating a particular $L(x)$ from all other physical systems, it must justify ascription of the function of implementing $L(x)$ to systems that cannot be singled out for their unique capacity to do so. If it is to succeed in this regard without trivializing the notion of computational capacity, it must furthermore provide the means to meaningfully quantify "how well" non-ideal artifacts implement specific logical transformations $L(x)$. Below I extend the ideal L-IPA theory of Sect. 3 to addresses these challenges, drawing from Hughes' characterization of artifact reliability and effectiveness (Hughes 2009) and Anderson's generalized L-machines (Anderson 2010).

4.1 Failure, Malfunction, and Normality

In his characterization of unreliable or ineffective artifacts—what we here call non-ideal artifacts—Hughes distinguishes failure from malfunction as follows (Hughes 2009):

> **Failure**: A token t *fulfills its function* in a particular application just in case the functional goal φ was realized as a result of that application. Otherwise, it *fails* to do so.
>
> **Malfunction**: A token t is *malfunctioning* with respect to a proper function if it is unable to *reliably or effectively* realize φ in some situations c satisfying C when used according to α, i.e. if $\alpha(t)$ is not a reliable or effective means to φ in such situations.

Note that failure is an all-or-nothing proposition, relevant to individual uses, whereas malfunction accommodates shades of grey in realization of the functional goal through the definitions of reliability and effectiveness (Hughes 2009):

> **Reliability**: The *reliability* of α as a means to φ in c is the probability π that, under conditions c, execution of the plan α will realize φ.
>
> **Effectiveness**: The *effectiveness* is the degree to which the functional goal φ would be realized as a result of α.

Reliability quantifies failure rate, presupposing a fact of the matter as to whether a functional goal is or is not achieved on each use of an artifact, whereas effectiveness more broadly quantifies the degree to which the functional goal of an artifact is achieved through its use. Hughes emphasizes the relative nature of these notions, taking a normal token of type T—a token "with all of the requisite features necessary to realize each of the functional goals of T in the manner intended" (Hughes 2009)—as the standard by which failure and malfunction are to be judged for tokens $t \in T$.

I now interpret these notions for generalized L-IPAs, specifying a normal token in this context and identifying measures that appropriately accommodate the multiple-input functional goals of L-IPAs. This involves choices of whether normal tokens, reliability and effectiveness should be defined for the device $\mathcal{D}$ of a generalized L-IPA or for the user-accessible composite $\mathcal{R}_{in}\mathcal{M}$ through which the user accesses $\mathcal{D}$.

Normality: I take the appropriate normal token for an L-IPA to be any realization of an L-IPA of type T_L that is, to an exceedingly good approximation, an ideal L-IPA of that type, or an appropriate stand-in for the same.[15] A token belonging to any subtype $T_L^{(k)}$ of an effectively ideal L-IPAs will suffice: after every use, the relationship between the input source $\mathcal{R}_{in}$ and the apparatus $\mathcal{M}$ will be identical for all ideal L-IPA realizations evaluating the same $L(x)$ for the same input. Thus, one can always compare the joint state of $\mathcal{R}_{in}\mathcal{M}$ for a generalized L-IPA to the joint state of $\mathcal{R}_{in}\mathcal{M}$ for an ideal L-IPA processing the same logical input x_i to see if the constitutive goal φ_i was achieved on that use. At this user level, there will always be a fact of the matter.

Reliability: I define L-IPA reliability at this same user level, where, on any use of an L-IPA token, the outcome y_j produced in the final readout either is identical to that of a normal token under the same input x_i or it is not.

A quantitative multi-input reliability measure defined at this level must satisfy two requirements: first, the measure must obviously reflect all possible inputs $x_i \in \{x\}_L$ if it is to capture an L-IPA's ability to reliability implement the multi-input function $L(x)$. Second, the measure must account for the likelihoods or relative frequencies with which the various inputs are applied in a use case of interest if, as in Hughes' reliability measure for single-input functions, it is to properly reflect the overall probability of success in a single numerical measure. Both requirements are met by the following multi-input reliability measure:

Reliability (multi-input): The *reliability* of α as a means to $\varphi = \{\varphi_i\}$ in c is the average

$$\langle \pi_i \rangle = \sum_i p_i \pi_i \tag{3}$$

of the probability π_i that, under conditions c, execution of the plan α will realize the constitutive goal φ_i, where p_i is ith input probability.

This measure, specialized to L-machines, has the form of the success probability measure used widely for characterization of noisy and faulty logic circuits. Note that $\langle \pi_i \rangle$ depends explicitly on the choice of readout measurement selected for the **Read**(y) step of the use plan, and also on the set $\{\pi_i\}$ of use-case-specific input probabilities.

Effectiveness: Defining the effectiveness of an L-IPA is a more subtle matter. At the user level, where there are only discrete inputs and outputs instantiated in $\mathcal{R}_{in}\mathcal{M}$, constitutive goals are either met on any particular use or they are not; there is no middle ground between success and failure and thus perhaps no apparent need to consider effectiveness. However, at the level of the device $\mathcal{D}$—where computational capacities are displayed—there *are* varying degrees to which device state transformations contribute to the achievement of functional goals. They are simply obscured by the measurement processes that generates and registers only discrete outcomes—outcomes that are intended to register the correct logical outputs but can in practice can be correlated to transformed states of $\mathcal{D}$ to varying degrees and in non-obvious

[15] A physical instantiation of a lookup table could, for example, stand in as a normal token for an ideal L-IPA.

ways.[16] Because of its implications for the definition of L-IPA effectiveness, this crucial feature of measurement processes warrants further discussion.

In the earlier discussion of ideal L-IPAs (Sect. 3), readout measurements were assumed to be selected so their outcomes simply reveal the identities of evolved device output states and do so without ambiguity. The existence of such measurements is guaranteed by the presumed distinguishability of the various output states $D_j^{(out)}$ in ideal L-machines, and such measurements are implicitly assumed to be used for readout. With this assumption, all computational work done by the L-IPA is necessarily done in state transformations of the device $\mathcal{D}$, with measurement providing nothing more than a computationally benign bridge. Measurement neither adds nor subtracts from the device's contribution to realization of the functional goal.

In the general case, however, there exist no physically possible measurements that can perfectly distinguish transformed device states. When transformed device states corresponding to different logical outputs are not mutually distinguishable because of noise and/or quantum effects, then the input-output relationship instantiated in $\mathcal{R}_{in}\mathcal{M}$—the relationship evident to the user—will not mirror that of *any* deterministic logical transformation. Computational capacity becomes a relative notion—devices display capacities for implementation of various computational functions to varying degrees. Furthermore, whether the transformed device states are perfectly or imperfectly distinguishable, there generally exist measurements that do computational work left undone by a device *or* that incompletely reveal computational work that was done by a device. At one extreme, there exist measurements that can produce outcomes corresponding to the correct logical outputs in L-IPAs based on devices that produce completely unfaithful output states—e.g. that simply implement the identity operation—by doing the required computational work in the measurement step of the use plan that was not done in the device evolution stage. At the other extreme, radically non-optimal measurements can produce completely random outcomes even when the device has flawlessly completed all of the desired computational work in the device evolution stage and the transformed device states are perfectly distinguishable from one another. All intermediate shades of grey between these extremes are, of course, generally possible.

Thus, in non-ideal L-IPAs, state transformations at the device level contribute to the achievement of L-IPA functional goals to varying degrees. Device-level capacities for implementation of a given L, themselves imperfect, can be mirrored, enhanced, or diminished by the readout measurement that bridges the device and user levels. While measurement is an essential ingredient for *use* of a device $\mathcal{D}$ for computation, and choice of measurement plays a non-trivial role in user achievement of computational goals, this role is generally far more complex than a benign and unambiguous reading of the results of computational processes already completed

[16] There can be cases where achievement of constituent functional goals cannot be considered a fact of the matter on any given use at the device level. If distinguishable input states are transformed into quantum-mechanically indistinguishable output states, as may be the case in nanoscale realizations, few interpretations of quantum mechanics would presume any such fact of the matter prior to readout measurement.

by the device. A theory of L-IPAs should recognize as much while accounting for the contribution of the device proper toward achievement of the functional goal of evaluating L. This motivates definition of L-IPA effectiveness at the level of the device $\mathcal{D}$, independent of the measurement processes selected to bridge the device and user levels. The necessary ingredients can be found in the theory of generalized L-machines (Anderson 2010).

With this, I introduce generalized L-machines and computational efficacy measures that quantify their inherent capacities for implementation of L. I then extend the theory of ideal IPAs developed in Sect. 3 to non-ideal L-IPAs based on generalized L-machines, which includes appropriate artifact reliability and effectiveness measures.

4.2 Generalized *L*-Machines and Non-ideal *L*-IPAs

A generalized L-machine is an M-input machine $\{\mathcal{D}, \{D_i^{(in)}\}, \Lambda\}$ supplemented by an M-input, N-output L-referent $\mathcal{R}_L$ (Anderson 2010). The L-referent is a physical instantiation of the transformation structure associated with L, such as a normal token of an ideal L-IPA that implements an M-input, N-output transformation L, providing a physical standard by which degree of success in implementation of L by an imperfect device $\mathcal{D}$ can be gauged. Generalized L-machines differ from their ideal counterparts in that, even when the M input states $\{D_i^{(in)}\}$ are distinguishable from one another, the evolved states $\{\Lambda(D_i^{(in)})\}$ need not be mutually distinguishable nor must they faithfully represent logical output states of *any* deterministic logical transformation. Thus, in generalized L-machines, there is no objective basis for associating a unique logical transformation L with the evolution operation Λ or for unambiguously associating particular logical outputs with evolved states $\Lambda(D_i^{(in)})$ as is the case in ideal L-machines. The best one can do in the general case is to quantify *how well* a generalized L-machine implements a function L. Appropriate quantitative measures are discussed below.

The inherent contribution of the device $\mathcal{D}$ to the evaluation of L in generalized L-machine—unobscured by the role of the readout measurement employed in any specific use plan—is quantified by two computational efficacy measures: the computational fidelity and the representational faithfulness. Formal definitions of these physical-information-theoretic measures are provided in Anderson (2010); here I discuss their meaning and connection to computational capacity to motivate their adoption as device-level L-IPA effectiveness measures.

According to the notion of computational capacity employed here, an L-machine displays the capacity to implement a logical function $L(x)$ to the extent that the transformation of physical input states $D_i^{(in)}$ to physical output states $\Lambda(D_i^{(in)})$ mirrors the abstract mapping of logical input states x_i into logical output states $L(x_i)$. Ideally, the mirroring of abstract logical transformations by physical state transformations implies two conditions: first, it implies that evolved device states associated with different logical outputs[17] should be *perfectly distinguishable* from one another. Second, it implies that evolved device states associated with the same logical output should be *exactly similar* to (or *perfectly indistinguishable* from) one another.[18] Both conditions are met in ideal L-machines, as discussed in Sect. 2.1, where the capacity of a device $\mathcal{D}$ for implementation L is unambiguous: every evolved machine state is either exactly similar to or perfectly distinguishable from every other evolved machine state. Either or both of these conditions is, however, imperfectly met in a generalized L-machine, where various evolved machine states are similar to or distinguishable from one another only to some intermediate degree. The computational fidelity and representational faithfulness measures individually quantify the degrees to which the first and second conditions are met in generalized L-machines,[19] capturing and quantifying shades of grey in these two complementary aspects of physical capacity for implementation of a logical function L. They are, for this reason, appropriate as graded measures of computational capacity and of device-level effectiveness for generalized L-IPAs.

A final note is in order regarding the notion and quantification of computational capacity—and thus of L-IPA effectiveness—that I have adopted here, which is straightforward in one respect and perhaps counterintuitive in another. It is straightforward in that it associates capacity with some reflection of computational state transformations in physical state transformations, which would seem—in some form or another—to be an essential ingredient of any physically grounded notion of computational capacity. One may wonder, however, if the present view of capacity—which regards both fidelity *and* faithfulness as essential aspects of computational capacity—is overly restrictive. Consider, for example, a machine that evolves input states into distinguishable physical representations of both the logical output of a some surjective function $L(x)$ *together with* a copies of the logical input x.

[17] Here, we say that an evolved device state $\Lambda(D_i^{(in)})$ is "associated with" logical output y_j if $y_j = L(x_i)$, where $D_i^{(in)}$ is the initial device state representing the logical input x_i.

[18] Two (generally statistical) evolved machine states $\Lambda(D_i^{(in)})$ and $\Lambda(D_{i'}^{(in)})$ are *perfectly distinguishable* if there exists *any* measurement consistent with physical law that could distinguish them perfectly, which is to say that they occupy disjoint regions of the machine's physical state space. The two states are, on the other hand, *exactly similar* (or *perfectly indistinguishable*) if there exists *no* measurement consistent with physical law whose outcomes could distinguish them in any way, which is to say that they are the same statistical state.

[19] The computational fidelity $\mathcal{F}_L$ quantifies the *distinguishability* of device output states evolved from input states that are associated *different* logical outputs. This selective physical state distinguishability is reflected in the amount of information about the output of an L-referent—the "correct output" for a normal L-machine token—that is in the evolved device states, as captured via a physical correlation entropy measure. The fidelity is normalized to the amount of correlation achieved for an ideal L-machine, so $0 \leq \mathcal{F}_L \leq 1$ with $\mathcal{F}_L = 1$ only when the first condition is met (as it is in ideal L-machines). The representational faithfulness f_L, on the other hand, quantifies the *similarity* of device output states evolved from input states that are associated with *the same* logical outputs. This selective physical state similarity is reflected in the average lack of information about the input of an L-referent—a record of the input state—that is in the evolved device states associated with each logical output. The faithfulness is also based on the correlation entropy and properly normalized with reference to L-machines, so $0 \leq f_L \leq 1$ with $f_L = 1$ when the second condition is met (as it is in ideal L-machines). Both $\mathcal{F}_L$ and f_L account for all possible inputs, appropriately weighted by the input probabilities relevant to a given use case. See Anderson (2010) for details and elaboration.

The present view of computational capacity would take the machine to be a non-ideal implementation of $L(x)$, since transformation of the physical states by such a machine is bijective and thus *does not* mirror transformation of the logical states by the surjective function $L(x)$. Noting that the machine *does* generate the the correct output of $L(x)$—leftover copies of the inputs notwithstanding—one may reasonably argue that the machine *does* unambiguously display a perfectly good capacity for implementation of $L(x)$ and thus that faithfulness is not an essential aspect of computational capacity.[20]

Comprehensive recognition of computing machines as physical systems reveals strong physical grounds for recognizing both fidelity and faithfulness as necessary ingredients of computational capacity. By the present view of capacity—which again reflects how well computational state transformations are mirrored by physical state transformations in a physical device—the machine described above is indeed a non-ideal ($f_L < 1$) implementation *of the surjective function* $L(x)$. However, it *does* regard this example machine as a perfectly ideal implementation *of the bijective function* $L'(x) : x \to L(x) \circ x$ (where $\circ$ denotes concatenation), since it is this bijective function $L'(x)$—*not* the surjective function $L(x)$—that is perfectly mirrored in the machine's (bijective) physical state transformations. The notion of capacity endorsed here thus emphasizes logical-physical mirroring of transformation structure, with the generation of correct outputs contributing incompletely to the display of physical capacity for implementation of a specified logical transformation. This emphasis on transformation structure may very well seem arbitrary or unnecessary if computation is viewed exclusively in abstract terms. Why, after all, should the retention of leftover inputs downgrade the machine's capacity to implement $L(x)$ if the correct output is to be found in the final machine state? The physical answer is that information can be shed from a physical system only at a dissipative energy cost, so there are real physical differences between a machine's capacity to faithfully mirror $L(x)$ [to generate $L(x)$ while erasing x] and its capacity to faithfully mirror $L'(x)$ [to generate $L(x)$ while retaining x]. Full acknowledgement of computing devices as real physical systems thus favors a notion of capacity like that presented here and codified in the fidelity *and* faithfulness measures. Indeed, these two measures provably provide a direct physical link between the minimum dissipative cost of implementing a logical transformation $L(x)$ in a generalized L-machine and the machine's capacity for implementation of $L(x)$, with capacity regarded and quantified as the degree to which the abstract input-output structure of $L(x)$ is mirrored in the transformations of physical states by the machine (Anderson 2010).

With this, the pieces are in place to define generally non-ideal L-IPAs:

> ***L*-IPA (non-ideal)**: An L-IPA of type T_L and subtype $T_L^{(k)} \in T_L$ is a physical device $\mathcal{D}^{(k)}$ that is ascribed the instrumental function of evaluating an M-input, N-output logical function $L(x)$ and that, through execution of a specified use plan $\alpha_L^{(k)}$, realizes the functional goal of evaluating $L(x)$ with some reliability and effectiveness.

[20] The author thanks an anonymous reviewer raising this potential objection.

The key elements are as follows:

- **Relevant Physical Systems**: A physical device $\mathcal{D}^{(k)}$ of subtype k and two related physical systems that can be controllably coupled to $\mathcal{D}^{(k)}$: an input referent system $\mathcal{R}_{in}$ that physically instantiates the ith of M logical inputs x_i in the physical referent state r_i of $\mathcal{R}_{in}$, and a measurement register system $\mathcal{M}$ that physically instantiates the jth of N logical inputs y_j in the physical register state m_j of $\mathcal{M}$. An L-referent $\mathcal{R}_L$ is also presumed when a standard for physical quantification of reliability and effectiveness is of interest.[21]
- **Instrumental Function Ascription**: An ascription

$$\langle \varphi_L, \alpha_L^{(k)}, C^{(k)}, T_L^{(k)} \rangle_L$$

  of the function **Evaluate** $L(x)$—expressed as a set $\varphi_L = \{\varphi_i\}_L$ of M constitutive functional goals $\varphi_i : x_i \rightarrow L(x_i)$—to the L-IPA when used according to use plan $\alpha^{(k)}$ in specified contexts of use $c \in C_L^{(k)}$ (see Sect. 3.4).
- **Reliability**: The *reliability* of $\alpha_L^{(k)}$ as a means to $\varphi_L = \{\varphi_i\}_L$ is the average

$$\langle \pi_i \rangle = \sum_i p_i \pi_i$$

  of the probability π_i that, under conditions $c \in C^{(k)}$, execution of the plan α_L will realize the constitutive goal φ_i, where p_i is ith input probability (see Sect. 4.1).
- **Effectiveness**: The *effectiveness* with which evolution of $\mathcal{D}^{(k)}$ alone contributes to realization of the L-IPA functional goal **Evaluate** $L(x)$ is quantified by the computational fidelity $\mathcal{F}_L$ and representational faithfulness f_L. $\mathcal{F}_L$ quantifies the mutual *distinguishability* of physical output states resulting from inputs that "belong to"—i.e. that, by $L(x)$, would ideally map into—*different* logical outputs. f_L, on the other hand, quantifies the average *similarity* of physical output states resulting from inputs that belong to *identical* logical outputs.

This theory of non-ideal L-IPAs addresses the challenges associated with distinguishing computers from other physical systems that were outlined at the beginning of this section, at least for digital computing artifacts used to implement logical transformations. The *function* **Evaluate** $L(x)$ is unambiguously *ascribed* to an L-IPA—whether ideal or non-ideal—by an agent. The *capacity* for evaluating $L(x)$ is, on the other hand, objectively rooted in the design, structure, and composition of the physical artifact, but is generally limited and is not uniquely identifiable with any particular L. This decoupling of function and capacity—and recognition of their differing origins—addresses the one challenge that was not addressed by the ideal

[21] While L-IPAs necessarily include an input referent $\mathcal{R}_{in}$—a physical instantiation of the inputs that is external to the device—an L-referent $\mathcal{R}_L$ is not required for use of an L-IPA by an agent. In practice, the results of computations performed through the use of artifacts almost always go unchecked. This is to say that most users of nontrivial computing artifacts "fly blind", placing their trust in artifact reliability without actively verifying artifact efficacy during use.

L-IPA theory of Sect. 3: the accommodation of *L*-IPAs whose intrinsic computational capacities are not up to realization of their user-ascribed computational functions but that are unambiguously used by agents to perform these functions anyway. This decoupling grants agents wide latitude in the ascription of computational functions to physical systems—artificial and natural—with the recognition that success in realization of the ascribed function is ultimately constrained both by the physically rooted capacities of the artifact and by the manner in which the artifact is used. The theory includes provisions for quantifying contributions to success at the user and device levels through the reliability and effectiveness measures, respectively. Capacity and use requirements essential for reliable realization of ascribed functions, reflected in these measures, serve as sturdy bulwarks against trivialization. Trivialization of computational function, which could be a concern when functions are agent ascribed, is further resisted by Hughes conditions for instrumental function ascriptions to be true (see Sect. 2.2).

It remains to address the most obviously counterintuitive implication of the instrumental view of computation underlying the IPA theory of this work: that computers do not perform computations when they are not being used by agents for computation, even when they display the same computational capacities that enable such use. I address this issue in the following section.

5 Instrumental Computation: Capacity, Function, and Use

According to instrumental computation, physical objects do not have "intrinsic" computational functions. Objects may have intrinsic computational capacities, but computational functions are ascribed to objects by agents who wish to use them to achieve computational goals. Functions do not belong to artifacts independent of the agents who ascribe functions to them and use them accordingly. The role of computers in computing is thus as artifactual means to the computational ends of agents with computational goals: *Artifacts don't compute, agents compute.*

This view is in obvious conflict with the commonsense notion of computations as processes "performed" by artifacts themselves, rather than by agents who use them for computation. This commonsense notion—that "computers do the computing"—is deeply embedded in the way we think and talk about computation, both informally and formally.[22] We are, for example, much more likely to say that IBM's Deep Blue computer beat Garry Kasparov at chess than we are to say that IBM's engineers used the Deep Blue computer to beat Mr. Kasparov at chess. This seems natural enough. In most realistic scenarios involving use of a computing artifact by an agent, the artifact does vastly more of the heavy lifting than does the agent (and

[22] Even in formal accounts and technical discussion of implementation that accommodate user roles, authors routinely refer to computations as being performed by physical systems (e.g. Horsman et al. 2014; Teuscher 2014; Konkoli 2015). Computer engineers, whose concerns *necessarily* include provisions for use by agents (i.e. I/O), routinely do the same.

typically does far more than the agent could ever do).[23] This is certainly the case for Deep Blue: IBM's engineers and programmers presumably did not themselves have the capacity to evaluate the consequences of possible chess moves as efficiently or effectively as Garry Kasparov. Since Deep Blue "did most of the work", and since its use was decisive, it seems only natural to credit the win to the artifact.

By the present theory, however, the complex artifact called Deep Blue, when properly prepared (energized and programmed), has the capacity to "play" chess but was ascribed the function of playing chess by the IBM engineers who used it to beat Mr. Kasparov. This construes computation as an activity of agents with computational goals who use artifacts to extend their capacities for realization of these goals, not as an inherent function of artifacts that is automatically inherited from their capacities: *no agent, no function*. It even allows that an artifact not being actively used to execute a computation can display the associated capacity—evolving exactly as it would if it *were* being used for this purpose—without executing that computation. Display of computational capacity is necessary but insufficient for execution of a computation.

This construal of "computation" and its relationship to the associated artifact—a "computer"—can be illustrated through analogy with other familiar artifacts that extend the capacities of human agents. Consider, for example, the binocular devices mounted on the observation deck of the Empire State Building. Each of these "viewers" is an artifact that can be used by an agent—in her role as a "viewer"—to extend her own capacities for "viewing". We say that the agent is "viewing" the Statue of Liberty when she looks at it through the device—when she is using the (artifact-) viewer in her role of (agent-)viewer—and we regard the artifact as performing its function if its use enables her to better view the landmark. Without contradiction, however, we understand "viewing" to a role played by the agent and not by the binocular device. We do not say that the (artifact-)viewer is viewing the Statue of Liberty when the observation deck is closed and nobody is around, even though photons entering the "input" side of the device are focused to spatial points just behind the eyepieces exactly as they are when an agent is making use of it. Viewing is something that agents do, sometimes unaided and sometimes through the use of artifacts, but it is not something that artifacts do.

Instrumental computation recognizes computation as being of a similar nature. My "computer" is an artifact that I, in my agent role as a "computer", use to extend my computational capacities. I say that I am "computing" when I, as (agent-)computer, am doing computations—executing computational functions—through the use of my (artifact-)computer. I take my laptop to be fulfilling its computational function if, through its use, I am successfully realizing computational goals that I could not (or that I choose not to) achieve unassisted. But, when my laptop is doing nothing at my behest or anyone else's, nothing requires that my laptop is "computing" any more than the binoculars at the Empire State Building are "viewin" when

[23] Indeed, the term "computer" became predominantly associated with a type of artifact (a computing machine) rather a human vocation around the time that computing machines meaningfully extended their users' computational capacities.

nobody is looking through them. Of course, the controlled redistribution of electronic charges in my laptop and the concentration of photons behind the eyepieces of the viewer give these artifacts their respective capacities. But without agents who purposefully harness these capacities, a redistribution of electric charge does not amount to "computing" any more than a concentration of photons amounts to "viewing"—they are simply displays of capacity. Computing, like viewing, is something that agents do—sometimes unaided and sometimes through the use of artifacts—but not something that artifacts do. As Searle has put it[24]

> (N)othing is intrinsically computational, except of course conscious agents going through computations (Searle 1992).

and

> The brute physical state transitions in a piece of electronic machinery are only computations relative to some actual or possible consciousness that can interpret the process computationally. It is an epistemically objective fact that I am writing this in a Word program, but a Word program, though implemented electronically, is not an electrical phenomenon; it exists only relative to an observer (Searle 2014).

Intrinsic computation, as a claim that inanimate objects objectively perform computations independent of agents, seems comparatively difficult to defend. The key difficulty is identifying the source of would-be "intrinsic" computational functions that give this claim its force.[25] Without identification of an intrinsic source of computational functions, intrinsic computation can only be the claim that a system's dynamics can be given a computational interpretation. Even if the given computational interpretation is based on a rigorously articulated computational description of physical systems, and even if such a description provides compelling metaphors and useful models of the behavior of complex systems, a claim that a system's dynamics can be given a computational interpretation is much weaker than a claim that the system "does computations" or "is a computer"—i.e. that it intrinsically executes computational functions.

One might look for a defense of intrinsic computation in teleological functions. One could claim that a computing machine has the teleological function of performing a particular computation because it was designed and manufactured for that purpose, but this provides an inadequate defense even if it is correct. An agent might execute a given computational function with equal success using an artificial system designed for this purpose (e.g. a silicon integrated circuit) and a natural system appropriated for this purpose (e.g. a DNA Okamoto et al. 2004 or slime mold Adamatzky 2015 computer). Unless natural systems like DNA and slime mold

[24] This view is similar to views long expressed by Searle, but the overall view of the present work is far more restrictive in the role it grants interpretation in deterimining what amounts to computation in physical systems.

[25] Piccinini's mechanistic account stands out as a well developed existing account that *does* offer an explicit physical definition of computation and identifies an intrinsic source of computational function.

are claimed to have the teleological function of performing this computation, e.g. because they were evolved to execute this function, the source of the shared computational function must lie elsewhere.

Intrinsic computation is perhaps most easily defended if the distinction between capacity and function is ignored or denied. Indeed, accounts of computational implementation commonly equate "computation" with display of computational capacity. However, the attribution of computations to physical systems is problematic even if only computational capacities are considered. Real systems do not support capacities for implementation of unique computational functions, because of the nondeterministic nature of the underlying physical processes (see Sect. 4). If a system's capacities are not unique to a particular computation, and if there is nothing more to the implementation of computations than display of computational capacity, then real physical systems intrinsically implement vast numbers of computations.

There are reasonable and rigorous criteria for singling out particular computations for which a system displays the most natural capacities,[26] and their acceptance alone amounts to a rejection of pancomputationalism. However, satisfaction of such criteria does *not* amount to acceptance of intrinsic computation. The display of a system's capacity for implementation of a particular computation, even if unambiguous, does not distinguish implementation of that computation from the enabling spatiotemporal patterns of physical activity that underwrite this display of capacity. If such a distinction meaningfully exists, then there must be more to computation in physical systems than display of capacity. After all, agents *do* unambiguously use physical artifacts to implement particular computations.

Instrumental computation provides the required ingredient. The computational functions of artifacts originate in the goals and intentions of agents. Agents unambiguously *use* computing artifacts to implement particular computational functions—to compute—and they do so even when no particular computational function can be singled out for which the artifact unambiguously displays a unique capacity to implement.

To summarize, the view outlined here—instrumental computation—regards computational functions as freely ascribed to physical artifacts by users. It regards computational capacities—whether adequate to an ascribed function or otherwise—as objectively rooted in artifact design, structure, and composition. This goes beyond the obvious recognition that use of an artifact for computation requires both an artifact with computational capacities and a user. It explicitly recognizes that users with functional goals give artifacts their computational functions,[27] and denies that the display of a physical system's computational capacities alone constitutes implementation of a computation. On this view, the implementation of a computational function requires both intrinsic capacity and ascribed function—it is at the nexus

[26] See, for example, Joslin (2006), Anderson (2010), Konkoli (2015), and Millhouse (2017).

[27] This holds even for "intelligent" machines that can learn and operate autonomously, which are artifacts that are ascribed functions of learning and operating autonomously. Adaptive evolution of behavior in response to training data and accumulated memory of environmental influences is itself an ascribed function.

of physically rooted artifact capacities and the computational goals and intentions of users. Instrumental computation naturally recognizes the multiple realizability of computation, accommodating conventional manufactured computers and unconventional computers based on artificial and natural systems. Computing artifacts built on radically different physical substrates can share common computational functions, even when the origins of their computational capacities—and the requirements for harnessing these capacities—differ wildly. Such differences are rightfully ignored at the level of computational function, which is agent-ascribed and appropriately agnostic about the origins of enabling artifact capacities.

I emphasize again that instrumental computation applies only to computations executed by agents through the use of artifacts. Its denial of intrinsic computation in inanimate objects does not deny that agents themselves compute intrinsically, i.e. with their inherent cognitive capacities. In fact, the present theory has nothing to say about the "intrinsic" computational capacities or functions of agents, other than that they are necessarily of a different nature than the artifact functions considered here. As instrumental functions, the artifact functions defined here are ascribed to artifacts by agents but not to agents themselves. They presume that agents have their own intentions, goals, and capacities, but assume nothing about the nature or source of these intentions, goals, and capacities. Consequently, the theory of computational function developed in this work is neutral on questions of whether agent capacities are intrinsically computational and whether agents have computational functions in some other sense. Thus, any notion of *computational* agent function compatible with the present theory would require a pluralism about computational functions[28]: use of a computing artifact by an agent would necessarily involve multiple interacting notions of computational capacity and function. Perhaps the analysis of agent-intrinsic computation as conceived in computational theories of cognition, considered jointly with instrumental computation in the context of artifact use, can provide further insight into the respective notions of computation.

6 Conclusion

Computers, as we know them, are technical artifacts. Specifically, they are information processing artifacts (IPAs) that objectively possess physically rooted computational capacities and are used to execute agent-ascribed computational functions. Execution of a computation is a functional goal of an agent, which an agent may realize with their own cognitive capacities (if possible) or through the use of a computer. As such, computations achieved through the use of artifacts are more than the physical processes that enable their use for computation. Those physical processes can not themselves constitute computations, contrary to familiar habits of thought, to the notion of intrinsic computation, and to the claims of pancomputalism.

[28] This contrasts with efforts to seek a unifying notion of computational function that apples equally well to artifacts and their users (Vermaas 2009; Maley and Piccinini 2017).

So it is according to instrumental computation, the view of computation that underlies the general theory of IPAs I have developed in this paper. I started with two essential ingredients: a minimal physical description of machines that have capacities for implementation of logical transformations (L-machines) and a theory of artifacts that can account for the the ascription of computational functions to these machines (Sect. 2). I then considered computing artifacts based on idealized L-machines (ideal L-IPAs) in detail. I augmented and specialized the definitions of functional goals, use plans, contexts of use, and artifactual types in Hughes' theory of instrumental artifact functions as required to accommodate IPAs based on L-machines, and discussed distinctions between dynamical evolution, computation, and information processing that are relevant to this context (Sect. 3). Next, I considered computing artifacts based on generalized L-machines (non-ideal L-IPAs). Unlike their ideal counterparts, the capacities of generalized L-machines cannot be uniquely associated with deterministic logical transformations because of structural imperfections, physical randomness (e.g. fluctuations and noise), and physical indeterminism (e.g. quantum state indistinguishability). I showed how such artifacts can unambiguously be ascribed the function of executing deterministic computations that they do not have the capacity to implement reliably, and introduced efficacy measures that appropriately quantify how well they implement functions ascribed to them (Sect. 4). I finally examined essential distinction between artifact capacities and functions in instrumental computation, and showed how it clarifies the distinction between computers and other physical systems (Sect. 5). I have emphasized distinctive features of instrumental computation throughout the paper, and noted contrasting views, but did not attempt a systematic, point-by-point comparison with other accounts of computation in this work. My focus here has been on articulation of the IPA theory and the notion of instrumental computation.

The present work is unique in its commitment to exploring the view that computations are functional goals of agents—not just spatiotemporal patterns of physical activity—and to elaborating the consequences of this view. The key conclusion is that devices used by agents for computation—computers—are technical artifacts with intrinsic computational capacities but without intrinsic computational functions. The display of a physical object's computational capacities thus does not—*can* not—alone amount to implementation of a computation. I have offered a concrete account of what *does* amount to computation in artifacts—i.e. what is required by and involved in the the use of artifacts by agents to realize their computational goals—and of what it does mean for physical systems to have computational functions if they do not have them intrinsically. While I emphasized digital computing artifacts for concreteness, and elaborated the IPA theory only for a specific class of such artifacts (L-machines), instrumental computation provides a generic account of artifact-assisted computation and the IPA theory presented here could be similarly elaborated both for broader classes of digital computing artifacts and for artifacts that implement non-digital computations.

Instrumental computation is inherently compatible with emerging viewpoints in unconventional and natural computation, accommodating systems that would make for very unconventional computing devices as potential substrates for future computing technologies. It offers an inclusive but rigorous answer to the question of what counts as a computer—an answer that supports a liberalism about the ascription of computational functions to physical systems and at the same time enforces a conservatism about the physically rooted artifact capacities that are required for reliable execution of computational functions. Such a mindset seems essential for the open-minded but scrupulous consideration of radically unconventional computing technologies, such as that reflected the 'unconventional computation catechism' of Teuscher (2014).

Acknowledgements I am grateful to Gualtiero Piccinini, Jesse Hughes, Corey Maley, İlke Ercan, and two thorough and constructive anonymous reviewers for valuable comments on earlier versions of this manuscript. I also thank Mike Cuffaro, John Norton, and other attendees at the Twenty-Fifth PSA Biennial Meeting poster forum for insightful discussions on a poster presentation of this work.

References

Adamatzky, A. (2015). Slime mould processors, logic gates and sensors. *Philosophical Transactions of the Royal Society A*, *373*(2046), 20140216.

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

Anderson, N. G. (2017). Information as a physical quantity. *Information Sciences*, *415–416*, 397–413.

Anderson, N. G., & Piccinini, G. (2017). Pancomputationalism and the computational description of physical systems. *PhilSci Archive*, ID: 12812.

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

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

Horsman, C., Stepney, S., Wagner, R. C., & Kendon, V. (2014). When does a physical system compute? *Proceedings of the Royal Society A*, *470*(2169), 20140182.

Horsman, D., Kendon, V., & Stepney, S. (2018). Abstraction/representation theory and the natural science of computation. In M. E. Cuffaro & S. C. Fletcher (Eds.), *Physical perspectives on computation, computational perspectives on physics*. Cambridge: Cambridge University Press.

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

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

Joslin, D. (2006). Real realization: Dennett's real patterns versus Putnam's ubiquitous automata. *Minds & Machines*, *16*, 29–41.

Konkoli, Z. (2015). A perspective on Putnam's realizability theorem in the context of unconventional computation. *International Journal of Unconventional Computing,* 11(1).

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

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.

Maley, C., & Piccinini, G. (2017). *A unified mechanistic account of teleological functions for psychology and neuroscience*. Integrating Psychology and Neuroscience: Prospects and Problems.

Millhouse, T. (2017). A simplicity criterion for physical computation. *The British Journal for the Philosophy of Science*, forthcoming. https://doi.org/10.1093/bjps/axx046.

Okamoto, A., Tanaka, K., & Saito, I. (2004). DNA logic gates. *Journal of the American Chemical Society*, *126*(30), 9458–9463.

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

Piccinini, G. (2016). The computational theory of cognition. In *Fundamental issues of artificial intelligence* (pp. 201–219). Springer International Publishing.

Piccinini, G., & Anderson, N. G. (2018). Ontic pancomputationalism. In M. E. Cuffaro & S. C. Fletcher (Eds.), *Physical perspectives on computation, computational perspectives on physics*. Cambridge: Cambridge University Press.

Piccinini, G., & Scarantino, A. (2011). Information processing, computation, and cognition. *Journal of Biological Physics*, *37*(1), 1–38.

Putnam, H. (1991). *Representation and reality*. Cambridge, MA: MIT press.

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

Searle, J. R. (2014). What your computer can't know. *The New York review of books*. www.nybooks.com/articles/archives/2014/oct/09/what-your-computer-cant-know/.

Teuscher, C. (2014). Unconventional computing catechism. *Frontiers in Robotics and AI*, *1*, 10.

Turner, R. (2018). *Computational artifacts: Towards a philosophy of computer science*. Dordrecht: Springer.

Vermaas, P. E. (2009). On unification: Taking technical functions as objective (and biological functions as subjective). In *Functions in biological and artificial worlds: Comparative philosophical perspectives*, Vienna Series in Theoretical Biology, pp. 69–87.

Vermaas, P. E., & Houkes, W. (2006a). Technical functions: A drawbridge between the intentional and structural natures of technical artefacts. *Studies in History and Philosophy of Science Part A*, *37*(1), 5–18.

Vermaas, P. E., & Houkes, W. (2006b). Use plans and artefact functions: An intentionalist approach to artefacts and their use. In A. Costoll & O. Dreier (Eds.), *Doing things with things: The design and use of everyday objects* (pp. 29–48). Abingdon: Routledge.

Vissol-Gaudin, E., Kotsialos, A., Groves, C., Pearson, C., Zeze, D. A., & Petty, M. C. (2017). Computing based on material training: Application to binary classification problems. In *Proceedings of the 2017 IEEE international conference on rebooting computing (ICRC 2017)* (pp. 274–281). IEEE.