# When Physical Systems Realize Functions…

**Matthias Scheutz (matthias.scheutz@univie.ac.at)**
Department of Philosophy of Science
University of Vienna
Sensengasse 8/10
A-1020 Wien

**Abstract**

After briefly discussing the relevance of the notions "computation" and "implementation" for cognitive science, I summarize some of the problems that have been found in their most common interpretations. In particular, I argue that standard notions of computation together with a "state-to-state correspondence view of implementation" cannot overcome difficulties posed by *Putnam's Realization Theorem* and that, therefore, a different approach to implementation is required. The notion "realization of a function", developed out of physical theories, is then introduced as a replacement for the notional pair "computation-implementation". After gradual refinement, taking practical constraints into account, this notion gives rise to the notion "digital system" which singles out physical systems that could be actually used, and possibly even built.

**Keywords:** computation, implementation, computationalism, realization of a function, digital system, computer, computational practice, cognitive science, artificial intelligence

## 1. Introduction

One of the most controversial and still ongoing debates in cognitive science centers around the claim that minds are *like* computers, that mental states are *computational states*. If this were indeed so, then research programs like strong AI might actually succeed one day in modeling as well as explaining what mind is. The main problem, however, with the "computational claim on mind" (CCM) is that is still not quite clear what it means to be *computational*. Smith (1998), for example, lists at least five different construals of computation, each of which approaches the issue from a different intellectual position, focuses on different aspects of computation, and pursues different purposes (explanatory, practical, etc.). Many proofs of "equivalence" tie these various notions of computation together, i.e., if a function is computable according to any one formalism, then it is computable according to all of them. Hence, it is commonly accepted that the term "computational" may be interpreted in any of these senses. The particular choice (or the canonical formalism) will depend on whichever seems more practical or appropriate for the enterprise at hand (in this case, the study of cognition).

There are, however, strong arguments against this endeavor of explaining mind in terms of computation: some disagree with established notions of computation and argue that these notions will be of no help for CCM because they even fail to capture essential aspects of computation (e.g., intentionality, see Smith, 1996). Others, accepting them, show that some of these notions, such as Turing-computability, are too "coarse-grained"

to be suitable for cognitive explanations (e.g., Putnam, 1988, who proves that every ordinary open system "implements" every finite state machine without input and output, or Searle, 1992, who polemicizes that even ordinary walls can be interpreted as "implementing" the Wordstar program). The former line of attack concentrates on our misunderstanding of what everyday computation is all about, whereas the latter debunks our understanding of how an abstract computation can be realized in the physical—how a computation is "implemented".[1]

In this paper, I will argue that a satisfactory account of implementation in order to answer essential questions such as "What computation does a given physical system implement?" ("the implementation problem") or "Can the human mind be described computationally?" is still missing. In particular, a solution to the implementation problem is *conditio sine qua non* for CCM, otherwise CCM is not even well-defined. After presenting what I take to be the methodological defects of standard approaches to "implementation", I will outline a theory of implementation which not only overcomes these obstacles, but as a consequence also suggests different notions of computation.

The first part of this paper starts out with a brief review of the standard conceptions of computation in their historical context, setting the stage for the presentation of *Putnam's Realization Theorem*. This claim (that every finite state machine is realized in every physical system) points, in my interpretation, directly to the lack of a thorough understanding of implementation. The consequence of Putnam's theorem is devastating for computionalism as well as the foundations of computation, since it shows that every theory of computation is built on weak grounds (if not on no grounds whatsoever) without a *theory of implementation*. Most attempts to counter and overcome Putnam-like results in order to rescue standard notions of computation (such as "Turing machine computable") share the view that implementation is a function relating computational and physical states (e.g., Chalmers (1996,1997), Copeland (1996), or Melnyk (1996)). In my view, this "state-to-state correspondence view of implementation" (as I coined it) fails, however, to get at the heart of Putnam's construction.[2] A methodological analysis concludes the first part and suggests that a state-to-state correspondence view is not general enough to explain how abstract computations can be connected to concrete systems. Instead of assuming abstract computations and asking how they relate to the physical, it seems more promising to approach the implementation issue by starting with physical systems and their mathematical descriptions (which will eventually give rise to computations).

The second part takes up this suggestion and shows that it is possible to tackle the notion "computation" from a practical point of view (by looking at the devices that humans design, build, and use). Physical descriptions of these devices will lend themselves to the very general notion "realization of a function", a *precursor and amalgam* of the notions "computation" and "implementation". The notion of realization of a function, in turn, will become more and more restricted as practical and/or physical constraints (such as measurability, feasibility, etc.) are incorporated. Although this account is far from being a full-fledged theory of implementation, it can describe relations between abstract and concrete systems in terms general enough to subsume standard computational systems. These systems are viewed as being realized by special

kinds of physical systems, called "digitality supporting systems". The main virtue of this approach for any resulting theory of computation is, then, that the respective notions of "computation" and "implementation" are not, as otherwise commonly maintained, defined at a set-theoretic level. Rather, they are mathematical extractions obtained from behavioral descriptions of concrete systems, developed from a progression through various levels of abstraction, thereby never abandoning—and, thus, retaining up to the highest level—their close ties to the concrete world.

## *Part I:*
## *Computation + Implementation = ?*

### 2. Turing Machines and Implementation

What is commonly considered "computation" (Turing machines, Markov algorithms, universal grammars, recursive functions, PASCAL programs, etc.), has as one of its defining characteristics the property of being independent from the physical. In other words, computations are (normally) defined *without* recourse to the nature of the systems that (potentially) realize them. This prominent feature, the independence of computations from their physical realizations, has made computation an attractive candidate for functional explanations of mental processes in cognitive science.

Among the various computational formalisms, the Turing machine model has gained most attention, ever since Putnam (1967) gave birth to (Turing machine) functionalism by suggesting how computational states of these machines could be used for understanding mental states. His "multiple realizability argument"—there might be different physical realizations for one and the same cognitive organization—shows that type-identity is too strong a requirement to link the mental to the physical by utilizing the independence of abstract Turing machines from their physical realization. From then on, the separation of computation from what does the computing, the "computer", was considered a virtue, not a flaw. While in the philosophy of mind the nexus between the mental and the physical has been the fulcrum of debates (determining the division lines between type and token-identity theory, functionalism, eliminativism, etc.), the link between computations and computers was implicitly assumed to be unproblematic. The naiveté of this assumption seems quite surprising in light of CCM: if minds are taken to be computational, then the relation between computation and computer is of crucial importance in its role of explaining the relation between mind and body. Yet, the theory of computation has mostly been concerned with the relations of various computational formalisms to each other.

Before actual computers even existed (except if one wants to count as existent all the mechanical "calculators" such as Babbage's "difference engine", etc.), the forefathers of the theory of computation—Gödel, Church, Kleene, and Turing—had already laid its foundations: the class of recursive functions equals the class of λ-computable functions equals the class of Turing machine computable functions. These results were possible,

because what a computation computes is expressed in terms of functions from inputs to outputs; and using functions as mediators, different computational formalisms can be compared according to the class of functions they compute. The extensional identity of all these formalisms supports a famous thesis formulated by Church (and by Turing independently) that any of these mechanisms captures our intuitive notion of computation, that is, *what it means to compute*. Although this thesis cannot be proved in principle, it gains a lot of plausibility given the above mentioned equivalence results.

Of these different formalisms the Turing machine is the only one which directly links computation to a mechanical device, to something physical. Whereas all the others tacitly assume that such a link is possible (e.g., to explain human computational behavior in terms of λ-computable functions), it is the Turing machine that *via* equivalence results provides this necessary connection to the concrete. This gives further evidence for its central role in the philosophy of mind as well as in the theory of computation. But it also means that if the Turing machine model fails to withstand attacks regarding its implementation, then all the other formalisms are cut off from the concrete as well.

When Turing invented his model of "computation", he wanted to capture the human activity of "computing", i.e., the processes a person goes through while performing a calculation or computation using paper and pencil. By "abstracting away" from persons, scratch paper, etc., he claimed that all "computational steps" a human could possibly perform (only following rules and making notes) could also be done by his machine (see Turing, 1936). That way the imagined device called *Turing machine* became a model of human computing, an *idealized* model, to be precise, since it can process and store arbitrarily long, finite strings of characters. Note that the level at which the mechanism of a Turing machine is described lies above the "mechanical level of description of physical bodies". It is rather the same at which we describe the behavior of a person when he or she performs a computation.

Nevertheless, at the core of the concept "Turing machine" lies the idea of a *mechanical* device that can actually be *built* (the infinite tape being substituted by an arbitrarily extendable tape, for example), or that, when taken as an idealized entity, *exists* in Plato's heaven. Either way, the mathematical structure, i.e., the quintuple that is usually taken to be "the Turing machine", can be viewed as a mathematical model or description of that (ideal) physical device.

The mathematical model serves a twofold purpose: it is not only thought to be a mathematical model of the device "Turing machine", but also an adequate representation of a human person performing computations with paper and pencil following rules. But what exactly is the relationship between the formal structure (i.e., quintuple) and the physical system (i.e., human)? The answer to the "implementation problem" for Turing machines is certainly non-trivial. Not even the relationship between the mathematical structure and the device "Turing machine" is all that clear (e.g., how are abstract states related to physical states?). Only at a very high level of description, can a correspondence be established (one could, for example, define physical states to be the position of the tape head plus the internal state of the machine plus all the characters on the tape).

The same is true of the relation between a human computer and the device "Turing machine". At the level of (classical) fields, for example, the question "How exactly do tape head movement and printing on tape relate to hand movements and scribbling on scratch paper?" is probably not answerable (see the next section). Thus, only by involving metatheoretical considerations about the nature of mechanical possibilities could Turing argue that humans (following rules) and Turing machines have the same "computational" limitations. Note that physical models of Turing machines are subject to the same kinds of practical constraints that humans are—neither do they have arbitrary amounts of scratch paper/tape nor time at their disposal. Hence, the abstract mathematical structure is an idealization of and theoretical limit for both.

But how about systems that do not obviously behave in such a way as to give rise to mechanical descriptions? Could those systems implement Turing machines? And what about the physical system "brain"? Does the brain with its various electric inputs and outputs implement a Turing machine? Even though the problem of what humans can compute with paper and pencil following rules seems decided, the general problem of what a physical system can compute according to the notion "Turing computable" will be left open if there is no theory that explains how to relate physical systems to the mathematical structure "Turing machine".

## 3. Putnam's Argument

The independence of computations from the physical systems realizing them seems to be their strength and weakness at the same time. Although computational formalisms allow one to specify which function *f* is defined by a particular computation (in the sense that the computation takes values from the domain of *f* as inputs and delivers values from the range of *f* as outputs), they implicitly presuppose a notion of implementation, i.e., that it is understood what physical systems can actually "run" the computation. But, as has clearly been pointed out by Putnam (1988), no satisfactory theory of implementation is yet available, since *every ordinary open system is the realization of every abstract finite automaton*. Given the central role that abstract finite automata play in computer and cognitive science, this result is more than puzzling.

Putnam's proof of this counterintuitive theorem—I christened it "Putnam's Realization Theorem"—hinges crucially upon a very "liberal" formation of physical states/state types, namely an arbitrary (possibly infinite) union of "maximal states", assuming a field theoretic level of description:

> *"In physics an arbitrary disjunction (finite or infinite) of so-called 'maximal states' counts as a 'physical state', where the maximal states (in classical physics) are complete specifications of the values of all the field variables at all the space-time points".* [Putnam (1988), p. 95]

Regions in phase-space, that is, sequences of physical states, which in turn are taken to be real-time intervals during the "lifetime" of a physical system *S* (e.g., all maximal states of *S* from to 12:00 to 12:04 on October 22, 1997), are set in correspondence to sequences of abstract states (determined by the machine table of the automaton)—see figure 1. This

way state transitions are "causal" in the sense that they can be predicted from the laws of physics given the physical conditions on the boundary of *S* throughout its lifetime.

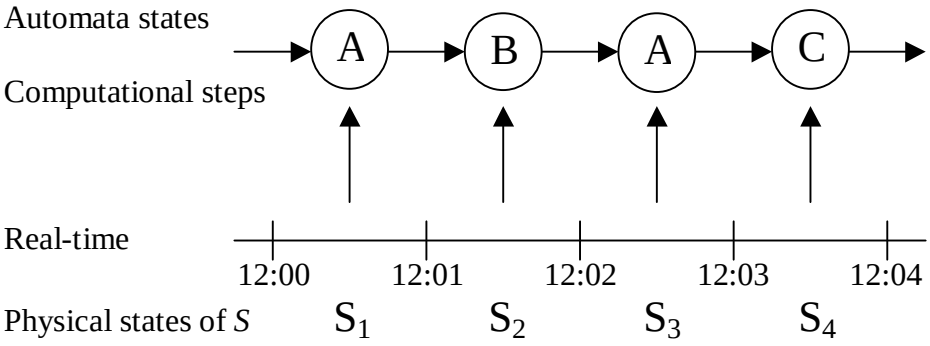


**Figure 1** Physical states are defined as sets of maximal states of a system *S* for a given interval of real-time (from 12:00 to 12:04) in such a way that they are in correspondence with automata states in a "run" ("ABAC") of the automaton.

*State types* are then defined as the union of all physical states corresponding to a single state of the automaton (note that these types are still considered "legitimate physical states" by the physical theory)—see figure 2.

Since "sequences" of state transitions (=computational steps in the abstract) are crucial to Putnam's construction, this will have to be reflected in a definition of what it means for a physical system to realize a finite state automaton (FSA):

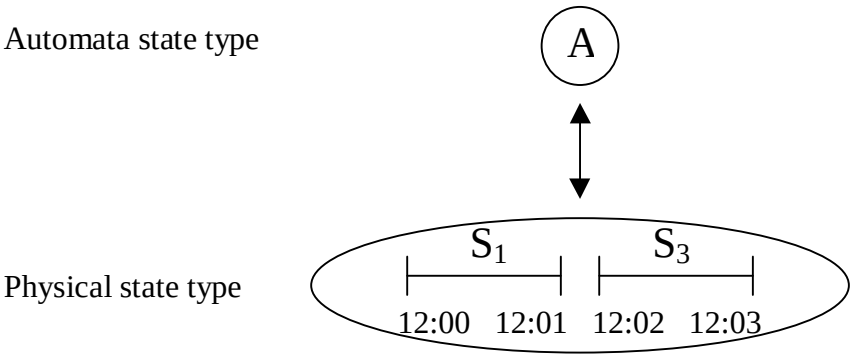


**Figure 2** Physical state types are defined as sets of maximal states of all physical states which correspond to the same automaton state resulting in an isomorphic mapping between physical and automaton state types—for details of the construction see the appendix of Putnam (1988).

*Definition 1:* A physical system *S* (described in a theory *P*) *realizes n* computational steps of a FSA (without input and output) within a given interval *I* of real-time if there exists a 1-1 mapping *f* from automata state types onto physical state types of *S* and a division of *I*

into *n* subintervals such that for all automata states *q*, *p* the following holds: if $q \rightarrow p$ is a transition in the automaton from the *k*-th to the *k+1*-th computational step ($1 <= k < n$) and *S* is in state *f*(*q*) during the *k*-th subinterval, then this will "cause" *S* to change into state *f*(*p*) in the *k*+1-th subinterval.

Given this definition, we can formulate Putnam's result:

*Theorem 2:* (Putnam's Realization Theorem) There exists a theory *P* such that for every ordinary open system *S*, for every finite state automaton *M* (without input and output), for every number *n* of computational steps of the automaton *M*, and for every real-time interval *I* (divisible into *n* subintervals) *S* (described in *P*) realizes *n* computational steps of *M* within *I*.

Putnam's diagnosis of what went awry to make his construction possible points to the liberal formation of physical states/state types. In order to avoid this kind of counterintuitive result we "[…] must restrict the class of allowable realizers to disjunctions of basic physical states […] which really do (in an intuitive sense) have 'something in common' "[Putnam (1988), p. 100]. In restricting the choices of physical states (that are supposed to correspond to computational states) to "natural" states which *really do* have something in common, one must not, however, involve "higher level" properties if the system is to *exhibit these properties* by virtue of its particular physical states; or in Putnam's words (who discusses the same problem for propositional attitudes, which are supposed to be reduced to physical-computational states):

> *"[…] this 'something in common' must itself be describable at a physical, or at worst at a computational level: if the disjuncts in a disjunction of maximal physical states have nothing in common that can be seen at the physical level and nothing that can be seen at the computational level, then to say they 'have in common that they are all realizations of the propositional attitude A', where A is the very propositional attitude that we wish to* reduce*, would just be to cheat."* [Putnam (1988), p. 100]

The main implication of Putnam's proof for the current enterprise is that without restricting the formation of physical state types a direct correspondence between physical state types and machine state types can always be found. Implementation viewed as such a mapping, however, cannot single out interesting physical systems as "computers", since the class of finite state automata (without input and output) coincides with all physical systems (describable at the level of physical fields) under this notion.[3] As a consequence, this result—if true—provides strong evidence against the tenability of a state-to-state view of implementation.

## 4. Problems with the State-to-State Correspondence View of Implementation

Assuming a notion of "computational state" (or abstract state, for that matter), there are basically two critical points if a direct mapping between abstract and physical states is to be established: either the mapping itself or the formation of physical states/state types

might not be constrained enough (or both). In Putnam's construction it seems that the state formation is the tricky part, whereas the mapping seems straight-forward. But interestingly enough, this alone is not what Putnam has been criticized for. Some find Puntam's notion of causality insufficient, others attack his (implicit) notion of "implementation". Chalmers (1996), for example, criticizes the correspondence between physical and abstract states insufficient. At the same time he assumes the formation of physical state types to be unproblematic. Similarly, Melnyk (1996) and Endicott (1996), too, in their endeavor to refute the arguments against computationalism in Searle (1992), hold a state-to-state correspondence view, which presupposes the formation of physical types. Without going into details of the respective arguments and presenting individual counter-examples, which I have done elsewhere (Scheutz, 1997), I rather prefer to advance a general argument that makes it seem very unlikely for a state-to-state correspondence view of implementation to succeed in general.

Physical states of an object are normally defined by the theory in which that object is described. As it happens with classical fields, there might be too many states that could potentially correspond to some abstract, in this case *computational*, state. In order to exclude certain unwanted candidates, one has to define an individuation criterion according to which physical states are singled out. This criterion, however, is not defined within the physical theory that is used to describe the object, but rather at a higher level of description. In the worst case, this will be exactly the computational level, namely in the case that none of the potential "lower level" theories can define a property in their respective languages such that the set of states conforming to that property corresponds in a "natural" way to the computational state. The potential circularity is apparent: what it is to be a certain computational state, is to be a set of physical states which are grouped together because they are taken to correspond to that very computational state.

Every state-to-state view of implementation must, therefore, avoid being 1) vacuously broad (because physical state type formations are too liberal), and 2) circular (because individuation criteria for physical states are not provided at a level lower than the computational one).

In the case of physical fields, one is left with a very pessimistic prospect: there are more than countably many different possible physical states according to the state space of fields (for every interval of real-time). Which of those correspond to a physically possible object, and which correspond to a given object in a "natural way"? Since there are *even more possible mappings* from physical states onto abstract states, it seems totally implausible if not impossible to specify finite criteria that single out the right mappings. The only way we could find such a mapping is either by pure chance or by using higher level properties that constrain possible objects significantly and hence the plethora of mappings. If we are lucky, then the number of mappings will be so constrained by these properties that we can actually write down the definition of a (correspondence-)function. But again, this "will work" only by using properties defined at levels of description higher than physical fields, yet lower than the computational level of description (which must not be used in defining a mapping from physical states to computational states, if the task is to find out what kind of computation a given physical system implements). One advantage of higher level theories is that they supply "higher level"-objects that can

be individuated according to criteria also supplied by these theories. Properties of these objects, in turn, could be used to define states.[4] State-to-state correspondences would then have to be defined separately for individual theories.

Involving higher order theories, however, does not solve the problem of forming physical state types if the theory does not provide such a concept. Take a pyramidal cell, for example, and its physical description in the language of biochemistry, which does not provide a notion of physical state type that could be set in direct correspondence with states of connectionist units. How would one go about defining physical state types such that the behavior of the cell corresponds to the computation of its "connectionist counterpart" and, at the same time, these type formation rules exclude type formations that would give rise to "unwanted" computations? This does not seem clear at all.

So the main difficulties of the state-to-state correspondence view of implementation sneak in the backdoor again, if the question "How are abstract computations tied to the concrete?" is asked. Even the rephrasing "What computation is implemented by a concrete system?" is not sufficient, since it, too, assumes a notion of computation. Because computations have to be linked to concrete systems (which are described at a certain level) in order to *be computations*, the implementation-relation must hold between computations and levels of descriptions of systems.

This leaves various questions unanswered: which (lower) level is the *right* one? Which level supplies the right kinds of states to be linked to the computational ones? Is there a systematic way to 1) find the right level and 2) find the right states/state types at this level? A theory of implementation should be able to answer all these questions in a systematic way for all possible levels of description. Any state-to-state correspondence view, however, is naturally limited to a level of description and a notion of state at that level (if it exists at all, otherwise the particular choice has to be justified with all its consequences…), and can, therefore, not provide any criterion for particular choices of levels. Furthermore, a theory of implementation should provide *necessary and sufficient* criteria to determine whether a class of computations is implemented by a class of physical systems (described at a given level), otherwise the term "implementation" is not appropriate. As long as these problems are not solved, Putnam-like constructions will present a potential threat to any state-to-state correspondence view of implementation, since there exist levels of description at which—paraphrasing a famous dictum by Feyerabend—*anything computes*.

## 5. Taking the Physical Seriously

The main reason for all these difficulties with a satisfactory account of implementation is, in my view, that computation has been defined abstractly at a "level of symbol manipulation", rather than in terms of an abstraction over the physical properties determining the functionality of a physical device. The latter approach is taken by computer practitioners, who have to define programming languages for the hardware they construct (in order to make it accessible and, hence, usable for other people). By abstracting over hardware specifics such as particular brands of parts, speeds of gates, etc. they are able to come up with an abstract description of what it means to *compute on*

*their kind of machine*. This way implementation and computation are defined together and the question of how computations are tied to the physical in general does not arise. It is this kind of practical wisdom that theories of implementation have yet to capture.

Even if one willingly granted Turing machines such a link (assuming that there are physical systems that correspond to them, ignoring all the aforementioned difficulties), they would still be mere models of what can be done *mechanically* (by a human, a robot, etc.). Gandy (1980), for example, shows that four principles underwrite the concept of "mechanical doability" and that a violation of each of these principles gives rise to "Super-Turing" computation. It follows that any system whose behavior can be completely described at the level of configurations, changes of configurations, etc. and which conforms to these four principles, will be at best Turing-computable. It does not follow, of course, that this level of description is the appropriate one for what humans can do (if they use scratch paper, but are not bound to using rules, say).[5]

The relevance for cognitive science (assuming CCM) is immediate: suppose brains are best described at a lower-than-mechanical level of description *L* (e.g., a biological level), and the mechanical level of description is not sufficient for *L*. If some of the phenomena not describable in terms of "mechanics" are crucial to a theory of mind, then either minds are not computational (if computational is meant to be "mechanical") or a different notion of computation is required (e.g., if one wants to describe biological systems such as cells, autopoetic systems, neural networks, etc. as "computational").

Considerations of this sort have already inspired many cognitive scientists to shift their explanatory framework from computational to dynamical, because they believe that the computational level of description is not essential to understanding cognition (see, e.g., van Gelder, 1998). Although one has to be careful with statements like this, because their truth depends on what "computational" means, I would agree that Turing machines are not well-suited to describe the behavior of various physical systems at lower levels such as chemical levels or even biological levels. And since the class of functions computable by Turing machines is the same as the class of recursive functions, it follows that these functions, too, *might* not be adequate to describe the input-output behavior of systems at lower levels of description. It even seems possible that the behavior of some physical systems could only be adequately described using recursively enumerable functions (certain quantum processes, say). Furthermore, if it were possible to utilize their behavioral complexity for "computational purposes", (computer) scientists would willingly extend their notion of "computation" to the class of functions "implemented" by those systems (see also the discussion section at the end of this paper).

All of these considerations together have led me to believe that a different theoretical framework is required in order to capture not only accepted, but also potential notions of computation. This formalism should allow one to define a corresponding notion of implementation (for each notion of computation) which explains how to link the abstract to the concrete without opening doors to Putnam-like constructions. One possible strategy to develop such a framework—the one I will take up in this paper—is to start in the concrete, in the physical, and not in the abstract: take a physical theory *P* and consider its (simple and complex) objects together with their properties. *P* will supply laws that describe the behavior *F* of a given arrangement of these objects—called physical system

*S*—under certain environmental conditions over time. Depending on *P*, objects, behaviors, and environmental conditions will be very different. However, every object in *P* will be subject to a certain change in some physical dimension (its "output") if exposed to certain environmental conditions (its "input"). The second part of this paper will show how input, output, and behavior of a physical system as described by a physical theory can be used to distill a (matter-independent) mathematical mapping *f—the function realized by S*—between inputs and outputs of *S* by abstracting over every physical dimension. This abstraction will not be arbitrary, but determined by *P*. Precisely because the trace to the "real stuff" is not forgotten, Putnam-like constructions cannot be applied when the notions "computation" and "implementation" (introduced for systems describable by physical theories) are later derived from the more fundamental notion "realization of a function".

## *Part II:*
## *? = Realization of a Function*

### 6. Setting the Stage: Electromagnetic Fields and Circuit Theory

Let us, then, start with a physical theory at a rather low level of description, *the level of electromagnetic fields*, and see how we can develop a notion of what it means for a physical system (described at this level) to realize a function. The theory of electromagnetism, as axiomatized by the four Maxwell equations, describes the behavior of (moving) charges: how (moving) charges give rise to two kinds of fields, the electrostatic and the magnetostatic field, and how these fields interact over time, resulting in electromagnetic fields (here, I assume classical fields for simplicity sake. I am not concerned with the integration of electromagnetic fields into quantum field theory, e.g., how particles arise out of fields, etc.).

In the simplest case, fields are studied in a vacuum, but Maxwell's equations can be modified to account for fields in material media as well as the change across material boundaries. In particular, the theory explains the interaction of potentials and currents in spatial regions "filled" with various materials over time (e.g., a cylindrical region of space filled with copper). Dividing materials into two rough categories, conductors and insulators, one can launch an investigation into the nature of different spatial combinations of materials giving rise to different electric properties. Without providing a detailed mathematical derivation, one can imagine how a categorization of combinations of different materials into types with the same electric properties could be attempted (by abstracting over particular material properties such as conductance or spatial arrangement such as volume). A cylindrical region filled with a given material in a vacuum, for example, will exhibit certain law-like properties with respect to the difference in potential between its two ends, if current flows through it. Furthermore, it will be possible to extract laws from the study of different such arrangements (e.g., Ohm's law, which

describes the relation between potential, current, and conductance of such "material regions").

This abstraction process leads, in the end, to the development of *circuit theory*, which arises from the theory of electromagnetic fields by abstracting over material properties of spatial regions as well as the regions themselves. It considers "closed-loop" arrangements (i.e., *circuits*) of two sorts of "higher level" objects: active and passive components. Active components are energy sources (e.g., batteries), passive ones are energy consumers (e.g., resistors, capacitors, or inductors). The "electric" properties of components and circuits are expressed in terms of "potential", "current", "conductance", "inductance", "resistance", "capacity", "voltage", etc.[6]

The reason for choosing circuit theory as a venture point in the current enterprise is twofold: on the one hand, circuit theory is motivated by computational practice (although it also has some relevance for the neurosciences—see the end of this section), since computers by and large are built out of electric components. On the other hand, it provides "basic objects", which can be individuated according to their properties and combined to form circuits, as opposed to electromagnetism, where space points together with their charge are the "basic objects" of the theory. Although one can *reduce* "circuit talk" to "field talk", a field-theoretic approach would distract from the current goal because of its mathematically involved nature. Additional information about certain kinds of materials, their atomic make-up, as well as facts from crystallography, atomic physics, chemistry, etc. would be needed to describe circuits completely at a level of fields. Circuit theory allows one to abstract over these "physical peculiarities" and assume objects such as resistors and capacitors without having to know their physical realization. Yet, one is guaranteed that these types of objects (within practical limits, of course) are readily available, i.e., can be built, since circuit theory was developed under the pressure of practical, engineering tasks. It, thus, combines the "physical rigor" of classical fields with the "engineering view" of idealized circuits, which are both necessary to describe actual and possible objects that are metaphysically tenable and physically plausible.[7] For the rest of this paper, I will assume circuit theory for all examples, yet allow "*P*" to range over any physical theory in all of the formal definitions, where the term "physical theory" is meant to comprise every theory that describes natural phenomena at a certain level of description in terms of "physical" laws (such as biology, chemistry, etc.).

Any (physical) theory used to describe real-world phenomena is built upon the mathematical framework that has been developed to describe *change*: the theory of differential equations (this is a matter of fact, not a necessity). Formally, this means that the theory consists of all mathematical (and logical) axioms needed for the theory of differential equations together with all necessary physical *eigenaxioms* (i.e., the axioms of the physical theory). It will contain additional predicates for different physical magnitudes (such as mass, energy, and time, for example) as well as other factors depending on the theory and the purpose it is used for. A formal version of circuit theory, in particular, could contain predicates like "is_a_resistor($x$)", "is_a_voltage($x$)", etc. and relational primitives like "has_resistance($x$,$y$)", "is_connected_to($x$,$y$)", etc. Using these predicates, one can formulate general laws such as

$$\forall x(\text{is\_a\_resistor}(x)\rightarrow\exists!y(\text{is\_a\_resistance}(y)\wedge\text{has\_resistance}(x,y))).$$

Given this axiom, a function symbol can be introduced for the resistance of a component, and this function in turn can be used to define Ohm's law. It should be obvious now, how I envision the process of defining a formal version of circuit theory, so I will move on.

## *7.* What it Means for a Physical System to Realize a Function

Physical theories, as already noted, describe the behavior of physical objects (and possible arrangements of them) under certain environmental conditions over time. What a physical object is depends on the theory under consideration. Each theory will supply a notion of primitive object (even if this can only be determined by looking at the domain over which the quantifiers of the theory range). I will use the term "physical system" to emphasize this theory-dependence. Circuit theory, for example, can describe the *behavior* of the system "resistor" when a voltage is applied. This can be also viewed as describing the "input-output function" of the component: take a 2 Ω resistor (the "component") to which a voltage of 200 mV (the "input") is applied. The current flow then—according to Ohm's law V=R*I (which is a special case of a differential equation where time is left out, i.e., $\Delta t$=0)—is 100 mA (the "output"). So, the resistor has the property of "embodying" the function $F(x)=x/2$ from voltages to currents.

What does it mean, therefore, to "embody" a certain function, to "have" a certain function, to "function" in a certain way? It means to *obey the laws of physics* that are described by that function. So, each concrete, individual resistor of 2 Ω will under an applied voltage of 200 mV yield a current of 100 mA—this is what the laws of physics predict (and if they are correct, this is what will happen modulo some practical problems such as "purity" of the material, exactness of the applied voltage, etc. which I will ignore for the moment). There are obviously critical notions involved in the above statement such as "predictions", "law obeying", maybe even "counterfactual", but I will not be able to go into details here.[8] However, exactly *how* the resistor "achieves" this mapping is a different question and can be answered by looking at the chemical structure, the arrangement of molecules, facts about electrons, etc. At this point we are only interested in the "*what*"-question and the answer to it does not require theories at "lower levels" of description.[9]

There is a sense in which the above resistor realizes the same function as a copper wire that is split into two parallel wires at some point: according to Ohm/Kirkhoff's laws the current will split in half (assuming they are joined together or fed into equal loads). So if the current at one end was 100 mA, then it will be 50 mA at each of the two other ends. The function "realized" by this wire is then $F'(x)=x/2$ from currents to currents, as opposed to the function $F(x)=x/2$ from voltages to currents. So, although the physical dimensions that are used for input and output (i.e., the domain and the range of the function that describes an aspect of the system under consideration) are different, the abstract mapping between those objects is the same. If, therefore, one drops the physical "qualities" (dimensions such as voltage or current) in the description of the "function" of

the components and just considers the "quantities" (magnitudes of the physical units), then one can describe the "input-output"-function of both components as $f(x)=x/2$ from numbers to numbers (*Reals* to *Reals*, say, given that voltages and currents are usually defined as *Real* values in circuit theory). In short: dimensions are dropped, units are retained.

This step of abstraction is critical to the whole enterprise and I will, therefore, develop the argument in greater detail. Eliminating physical particulars of inputs and outputs, will allow us to describe what resistor and wire have in common with respect to their mathematical (functional) description, namely that "the resistor and the wire both *realize* the function $f(x)$" (in different physical ways, of course). The resistor realizes *f in the sense* that for every voltage $x$, if $x$ is applied to the resistor, a current of $x/2$ will result by the laws of physics; the wire, in that for every current $x$, if $x$ is applied to one end of the wire, a current of $x/2$ will result by the laws of physics on each one of the other ends. What is, therefore, different is the domain of $f$ (since in one case it is the set of all currents, in the other the set of all voltages), what is the same is the syntactic structure of (the definition of) the function. There are two ways to express the difference between voltages and currents, 1) syntactically by using a two-sorted logic: $f(v)=v/2$ and $f(a)=a/2$ (where "$v$" is a variable ranging over volts and "$a$" a variable ranging over amperes), and 2) semantically by using a model-theoretic interpretation $\forall x\in V(f(x)=x/2)$ and $\forall x\in A(f(x)=x/2)$ (where "$V$" denotes the set of voltages and "$A$" the set of amperes). In the former case, the algebraic field axioms (within the formal theory) coincide for both sorts.[10] In the latter case, $V$ and $A$ can be shown to be extensionally identical up to isomorphism.

Given these two ways of capturing the difference between volts and amperes, one would like to arrive at the mapping $f(x)=x/2$ from *Reals* to *Reals*. One strategy is to argue that physics uses a mathematical language to describe concrete objects and that both volts and amperes are modeled by *Reals* in that language.[11] This opens the discussion about whether *Reals* are the "right models" for physical quantities, etc.

Another argument comes from model theory stressing the fact that two sets are *identical up to isomorphism* (with respect to the field operations "+" and "*"), if they satisfy all field axioms. It follows that a model where the set of Volts is the set of *Reals* is as good a model as one where the set of Volts is the set of *Rationals*, since the extensions of the predicates "$V$" and "$A$" are isomorphic (this is, of course, only true for the field axioms. If additional axioms are added, e.g., that every quadratic equation has a solution, then the *Rationals* might get excluded). This view depends on one's stance on issues like "standard interpretation" and "standard models" (and is related to questions like "How are the real numbers constructed?", "What are real numbers, sets of natural numbers or limits of Dedekind cuts?", or "What is the standard model of real numbers?").

A third possibility is to stay within the syntactic realm of the formal theory. Then one can treat variables of different sorts as being of yet another sort, namely the sort "*Real*", if all field axioms are defined for all of them. In the one-sorted case, one can substitute "$x\in Real$" for every quantifier restricted to a set (e.g., "$\forall x\in Real\ldots$" for "$\forall x\in V\ldots$"), if all field axioms are defined for that set. Formally, this means that the theory is either extended by another sort plus all the axioms for that sort or that a new (name for a) set is

introduced together with all relevant axioms. That way one avoids being forced to take a stand on either physical modeling paradigms or whether "identical up to isomorphism" is sufficient to "nail down" the set of *Reals* (i.e., if isomorphism is sufficient to determine a set of objects or if "more" is needed). Hence, I will use the term "syntactic isomorphism" in the following to emphasize that I have this third syntactic alternative in mind, even though I will treat isomorphisms semantically for the sake of expositional clarity. Note that this way even nominalists who are not committed to the existence of *Reals* should be able to accept the following definitions.

The "common structure" of the two functional descriptions $F$ and $F'$ for resistor and wire, respectively, can now be viewed as the mapping $f(x)=x/2$ from *Reals* to *Reals*. The "abstraction" over the physical dimension is achieved by supplying two syntactic isomorphisms: the *input encoding* $I(x)$ from voltages to *Reals* and the *output encoding* $O(x)$ from currents to *Reals*. Figure 3 depicts the relations among $F$, $f$, $I$ and $O$ (in this case for the resistor, but it really works for any physical system).

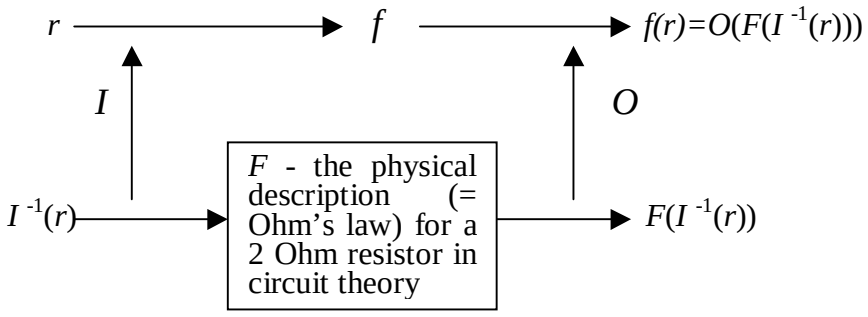


**Figure 3** The relation between the resistor and the function it realizes: given a certain *Real r*, the value $f(r)$ is then obtained by taking the *encoding of the input* $I^{-1}(r)$, applying it to the resistor, and then decoding the output $F(I^{-1}(r))$, using the *output encoding*, resulting in $O(F(I^{-1}(r)))$, which is equal to $f(r)$.

This suggests a general/generic definition of what it means for a physical system $S$ described by a (physical) theory $P$ to *realize a function f*:[12]

*Definition 3:* A function $f$ with domain $D$ and range $R$ *is realized* by a physical system $S$ (describable in a theory $P$) if and only if the following conditions hold:

1. There exists a (syntactic) isomorphic mapping $I$ from the "input domain" of $S$ to $D$[13]
2. There exists a (syntactic) isomorphic mapping $O$ from the "output domain" of $S$ to $R$
3. There exists a function $F$ that describes the physical property (=behavior) of $S$ for the given input-output properties (i.e., $F$ is a mapping from the "input domain" of $S$ to its "output domain" described in the language and by the laws of $P$) such that for all $x \in D$ the following holds: $O(F(I^{-1}(x)))=f(x)$.

Note that just requiring bijective mappings $I$ and $O$ is not enough to determine $f$: let $I(x)=x$ for all $x$ except for $I(1)=0$ and $I(0)=1$ (and the same for $O$), and $F(x)=x/2$. Then $f(1)=O(F(I^{-1}(1)))=O(F(0))=O(0)=1$, although $f(1)=1/2$ would have been the correct value.

One could even argue that "isomorphism" is still too weak, since it does not distinguish between $I$ and all $I'$ such that $I'(x)=aI(x)+b$ for all $a$, $b$ in the domain of $f$ (see also Cummins (1989), pp. 102). The same is true for the output encoding: $O'(x)=cO(x)+d$ for all $c$, $d$ in the range of $f$. Therefore, any one of the functions $f'(x)=cf(ax+b)+d$, i.e., $c(ax+b)/2+d$ (which reduces to $a'x+b'$ for $a'=ca/2$ and $b'=cb/2+d$) seems to be a possible candidate for the function realized by $F$. However, not all of them are reasonable, since $f'(x)=x$ for $a'=1$ and $b'=0$, for example, and this obviously defeats the purpose of the resistor. In a way, there is only one (isomorphic) mapping that can do the job, the one taking "0-volt" to "0", "1-volt" to "1", $n$-volt (where "n" is defined as the sum of $n$ "1-volt" elements) to $n$, etc. This also suggests an answers to a similar problem posed by Cummins, who introduced the notion "direct interpretation" for a particular isomorphism between symbolic and physical input/output, which he admittedly could not define (Cummins, 1989, p. 104).

The above definition, being cast as a *schema* (parameterized by the theory $P$) to allow for greatest possible generality, is naturally quite vague. It can neither specify what "physical system" means (let alone its behavior) nor what the inputs and outputs are for that system, but that is not its purpose anyway. These "empty places" will have to be filled in with particular values from the respective theory $P$ to make the definition complete. In the above examples, one would substitute "circuit theory" for "$P$", "resistor" for "physical system" together with the appropriate inputs and outputs (i.e., voltages and currents at the two ports of the resistor) as described. Other problems, however, such as the (short) time lag between the point of application of the voltage/current and the current on the output side (electrons moving only at a finite speed), or the range of functionality of the physical system, require attention and will be tackled as we start to mould this definition guided by practical constraints.

## 8. Analog Electric Circuits

So far, we have talked about the fact that electronic components can be described by an "input-output" function (which will naturally differ from component to component). Each individual function is rather restricted, hence quite simple. So in order to allow for more complex functions, one could consider more complex arrangements of these components (systems of components). Take, for example, high-pass filters (consisting of a resistor and a capacitor) which realize functions from frequency to frequency that will be the identity for high frequencies and the constant function $f(x)=0$ for low frequencies. One could also consider systems that change their input-output behavior depending on other external factors (e.g., resistors change their resistance depending on their temperature). In particular, these external factors might be exploited to make a system more versatile. Take, for example, an amplifier which contains a potentiometer to allow for adjustment of the amplification factor. It will then realize the function $f_p(x)=xp$ (where $p$ is the amplification factor, $0<p<=100$, say). Notice that this function is parameterized by and dependent on $p$.[14] These kinds of functions are especially practical, because they allow a single system to realize multiple functions; in other words, they make it a "multi-functional" system. Every radio, for example, is such a multi-functional

system, being capable of receiving multiple channels and extracting the low frequency information that was coded in the high frequency at different volumes (its function is parameterized by "volume control", "channel selection", etc.).[15]

Although all components work in the real world and are, therefore, subject to time and space constraints, I have not taken either into account. Take, for example, a delay circuit (which outputs incoming signals, i.e., voltages, after a certain delay *d*). This system seems to realize the identity function $f(x)=x$, but we would agree that it differs in an essential way from a single copper wire (which also realizes the identity function): the former only computes identity if time is left out; otherwise it realizes the function $f(x,t)=g_x(t-d)$ (where $f(x,y)$ is 0 for all $y<d$ and $g_x(t)$ is the function that describes the value of $x$ at time $t$). This function is much more complex than the identity function, which now can be seen as a special case when $d=0$ (obviously, if time is taken seriously, then no physical system will ever realize the identity function, since $d$ will never be 0).

Since time dependencies between input and output do not have to be constant either—just take a delay where the delay factor is a multiple of the magnitude of the input signal—even sophisticated input-output behaviors (with respect to time) are possible, once *time matters*. Thus, the definition of "realization of a function" has to be augmented by two time factors: an abstract time which is attached to the function *f* and the real-world-time as described by *P*. In the following "*RealTime*" will always denote the set of times derived from the theory *P* (e.g., the projection of the forth coordinate of space-time), whereas "*Time*" is supposed to denote the set of abstract times (intervals or points).

Another important factor is the physical condition of the system when input is applied, since, in most cases, the input-output behavior of the system will depend on it. The output of a fully charged capacitor for a given input differs significantly from its output for the same input if it is uncharged, for example. Hence, it is essential that the physical description of the behavior of a system also contain a (complete) description of its physical condition at the time when the input is applied. I will assume from now on that this description of the physical condition is part of the description of *S* (e.g., the initial conditions for a dynamic system).

*Definition 4:* A function *f* with domain $D \times Time$ and range $R \times Time$ is realized by a physical system *S* (describable in a theory *P*) if and only if the following conditions hold:

1. There exists a (syntactic) isomorphic mapping *I* from the "input domain" of *S* to *D*
2. There exists a (syntactic) isomorphic mapping *O* from the "output domain" of *S* to *R*
3. There exists a (syntactic) isomorphic mapping *T* from *RealTime* to *Time*
4. There exists a function *F* that describes the physical property (=behavior) of *S* for the given input-output properties over *RealTime* (i.e., *F* is a mapping from the "input domain"$\times$*RealTime* of *S* to its "output domain"$\times$*RealTime* described in the language and by the laws of *P*) such that for all $t \in Time$ and all $x \in D$ the following holds: if $F(I^{-1}(x),T^{-1}(t))=\langle y,r\rangle$, then $\langle O(y),T(r)\rangle=f(x,t)$ (for $y \in$ "output domain of *S*" and $r \in RealTime$).

This straightforward augmentation has wanted as well as unwanted effects. The input-output behavior of a system is now described as a functional relation between two graphs:

the graph of the input signal and the graph of the output signal over time. This way, the temporal behavior of a system can be completely described. Suppose, for example, that it takes time *d* for the electrons to pass a wire, then the function realized by the wire is basically the one described above for the delay. However, abstract functions now have a "time" attached to them and it is not quite clear what it means to have a "timed" version of the addition function, say. One possible answer is to argue that physical systems simply do not realize "timeless" functions (i.e., functions from timeless inputs to timeless outputs). In other words, every input to a physical system has to happen in time and hence the function which is realized by the system has to have a time parameter attached to it (e.g., a *Real* parameter). This point has been stressed by adherents of dynamicsm as one of the major shortcomings of the standard notion of computation: computations are defined in terms of *computational steps*, not in terms of *time* (e.g., see van Gelder, 1998).

In some cases this time parameter is exactly what distinguishes one system from another; given a delay circuit with 10 msec delay and another with 15 msec, dropping this parameter would mean no longer being able to tell the two systems apart.[16] Often, however, the time relation between input and output does not matter. Two wires, made of different material with different lengths, might (theoretically) still have the same resistance (except that the output is sooner available in one wire than in the other). In this case, we would like to ignore the time lag between input and output in order to be able to speak of "the same function that both realize". So, we first have to define what it means for a system to realize a "timeless" function:

*Definition 5:* A "timeless" function $f(x)$ with domain $D$ and range $R$ is realized by a system $S$ (describable in a theory $P$) if and only if the following conditions hold:

1. There exists a (syntactic) isomorphic mapping $I$ from the "input domain" of $S$ to $D$
2. There exists a (syntactic) isomorphic mapping $O$ from the "output domain" of $S$ to $R$
3. There exists a function $F$ that describes the physical property (=behavior) of $S$ for the given input-output properties over *RealTime* (i.e., $F$ is a mapping from the "input domain"$\times$*RealTime* of $S$ to its "output domain"$\times$*RealTime* described in the language and by the laws of $P$) and there exists a "delay-function" $d(x)$ from inputs to times (derived from $F$) such that for all $r \in$ *RealTime* and all $x \in D$ the following holds: if $F(I^{-1}(x),r)=\langle y, r+d(x)\rangle$, then $O(y)= f(x)$ (for $y \in$ "output domain of $S$").[17]

Notice that the time delay of the output is defined as a function of the input because the relation between time lag and input will not be constant in all systems, but may depend on specific inputs (e.g., if input and output to a capacitor are currents, then the capacitor will produce delayed output depending on its capacity). This definition can then be used to show that both wires discussed above, in fact, realize the same "timeless" function $f(x)=x$: there exist two delay-functions $d_1(x)$ and $d_2(x)$ (from inputs to times), namely $d_1(x)=0.01$ and $d_2(x)=0.015$ for all inputs $x$, such that if $x$ is the input to both systems at time $t$, then one system will output $x$ at time $t+d_1(x)$ and the other will output $x$ at time $t+d_2(x)$. The last step accomplishes a crucial abstraction: the actual duration, the link between input and output, the "behavioral" process that the system exhibits when an input is applied, resulting in a delayed output, is neglected in favor of a "timeless"

mapping! This way various different systems will realize the same timeless function independent of the time lag between input and output, i.e., their "speed". It now becomes possible to compare different systems with respect to their "functionality", when time does not matter. This, again, is a commonplace for computer practitioners: a PC with a 133 MHz CPU and one with 200 MHz CPU (other things being equal) differ only in speed, but not in functionality.

Three remarks seem necessary at this point:

First, physics certainly places restrictions on the domain of the input, the domain of the output, and the system itself. For example, one cannot expect to apply arbitrarily high currents to a wire of a given size, not only because it would be hard to generate them, but also because the wire would melt. This kind of dependence automatically delimits the range of the abstract function that a physical system is able to realize. So, strictly speaking, a wire that realizes the identity function according to the above definitions does not realize the *whole* function, but only a part of it (the part in which the wire "operates normally" according to the laws of physics; in the other part it will, of course, still work according to the laws of physics, but different factors will come into play, and the original equations will no longer apply). There are also physical reasons why certain components have to have (at least or at most) a certain size (e.g., a capacitor that can store the charge of 500 F will be too big to be soldered into a standard circuit board). These constraints, in turn, might have an influence on the time-factor of the system, as space and time are inseparably interwoven.

Second, there is also a limit to the accuracy with which an input signal can be generated and an output signal can be recognized (see also Haugeland, 1982). This problem results from the limits of measurability of physical magnitudes, no measurement will be 100% exact, but will always contain an error—i.e., will be within some (small) range of the actual value. Hence, from a practical point of view, the mere knowledge that a certain circuit actually realizes a very complicated function might not be of great help if there is no way of making inputs precise enough and/or reading off its outputs with sufficient precision.

And last, one final remark concerning the nature of electric circuits. When one hears "electric component", one automatically associates this term with man-made parts that are used to solder circuits. Thus, one implicitly assumes a certain physical structure: silicon-arsenide molecules, gold wires, etc. And, in fact, at the beginning of this section, I suggested exactly that. However, one could broaden one's perspective and also subsume "natural" (i.e., non man-made) *electric components and circuits*. In particular, one could view neurons as circuits with electric properties (a description of the functionality, of course, will include laws of chemistry, cell-biology, etc.). Given their electric properties and the nature of their inputs and outputs, neurons then realize certain functions (the ones that can be related in the above sense to their physical make-up). And the fact that neurons are made of "biologically describable stuff" does not mean that there might not be a man-made "artificial neuron" (possibly made out of inorganic substances) with the same properties (or if not entirely the same, then at least with respect to the input-output behavior) that in turn will realize the same functions. So one way to understand natural cognitive systems is to analyze what functions neural circuits realize,

at exactly the electric level of description. These functions can then be compared (with or without taking time into account) with the functions realized by artificial systems, or analyzed mathematically. In any case, it will be possible (once the physics of neurons is fully understood, if that is ever possible...) to determine the class of functions that are realized by neurons. And that, in turn, will eventually allow us to look at the complex input-output behaviors of networks of neurons (which then can be described as possible combinations—such as compositions, iterations, etc.—of the class of functions realized by a single neuron).

## 9. Digital Electronic Circuits

The last section showed that the electric level of description allows us to describe a vast variety of complex circuits together with the functions they realize. However, I have already mentioned that theory is only part of the story, and that practice is quite another. Although one can theoretically define devices that realize very interesting functions, it might not be possible to build and/or detect them (due to a lack of sufficiently precise tools, measurement instruments, etc.). Another factor to consider is the impact of the environment on real-world devices. The influence of noise levels and other disturbances might prevent the circuit from functioning according to the "idealized" laws of physics ("idealized" in the sense that all possible environmental influences are normally *not* taken into consideration in standard formulations of physical laws when used in practical settings...).[18] Hence, every description will have a "small" error term ε attached to it for the margin of error within which the system's behavior cannot be *exactly* predicted. However, once such an error boundary is given (together with reliable physical conditions that the system will stay with sufficiently high probability within these boundaries taking the environment of the system into account), then one can use these systems for an inquiry into the nature of the functions they realize. The main difference now (compared to the previous definition 5) is that the output of such a system will not be the same under the same input conditions, but always within a certain interval (actually, the same is true for the input as well, since it is practically impossible to generate arbitrarily precise inputs).

This change in precision (i.e., the relaxation of the constraints) has to be taken into account in the formulation of a new version of the definition "realization of a function". Note that four different error terms will have to be defined in advance ($\varepsilon_{din}$, $\varepsilon_{dout}$ for input-output magnitudes and $\varepsilon_{tin}$, $\varepsilon_{tout}$ for input-output times):

*Definition 6:* A "timeless" function $f(x)$ with domain $D$ and range $R$ is (practically) realized by a system $S$ (describable in a theory $P$) with $\varepsilon_{din}$, $\varepsilon_{dout}$ for input-output magnitudes and $\varepsilon_{tin}$, $\varepsilon_{tout}$ for input-output times if and only if the following conditions hold:

1. There exists a (syntactic) isomorphic mapping $I$ from the "input domain" of $S$ to $D$
2. There exists a (syntactic) isomorphic mapping $O$ from the "output domain" of $S$ to $R$
3. There exists a function $F$ that describes the physical property (=behavior) of $S$ for the given input-output properties over *RealTime* (i.e., $F$ is a mapping from the "input

domain"×*RealTime* of *S* to its "output domain"×*RealTime* described in the language and by the laws of *P*) and a "delay-function" $d(x)$ from inputs to times (derived from *F*) such that for all $r \in RealTime$ and all $x \in D$ the following holds (for notational convenience abbreviate the error interval for $y$ determined by $\varepsilon$ as $\mathrm{INT}(y,\varepsilon)=_{def}\{y|y-\varepsilon<=y<=y-\varepsilon\}$): if $F(I^{-1}(\mathrm{INT}(x,\varepsilon_{din})),\mathrm{INT}(r,\varepsilon_{tin}))=<Y,Z>$, then $O(Y)\subseteq\mathrm{INT}(f(x),\varepsilon_{dout})$ and $Z\subseteq\mathrm{INT}(r+d(x),\varepsilon_{tout})$ (for $Y\subseteq$"output domain of *S*").[19]

This definition works in a practical setting as follows: suppose the system *S* is given together with a physical description of its functionality *F* and measurement errors $\varepsilon_{din}$, $\varepsilon_{dout}$ for the input-output magnitudes, and $\varepsilon_{tin}$, $\varepsilon_{tout}$ for the input-output times. For every input $x$ to the system (which could be anywhere between $I^{-1}(x-\varepsilon_{din})$ and $I^{-1}(x+\varepsilon_{din})$), the output needs to be within $O^{-1}(f(x)-\varepsilon_{dout})$ and $O^{-1}(f(x)+\varepsilon_{dout})$ (under the given mappings *I* and *O*, of course). If *f* (or *F* for that matter) is unknown, then it can be approximately determined by repeatedly applying various values of $x$ to the system. Engineers, for example, when they measure voltages at different places in a circuit in order to find the reasons for the system's improper functioning, use implicitly a definition of the above kind. In general, every measurement will reach the limits of precision at some point, and then the "original" value can only be "estimated" from the results of many different measurements (within a certain interval).

Although this definition seems more appropriate given practical limitations, it is still not satisfactory. The most urgent problem is certainly one that cannot be accounted for by a definition in principle: what if the errors are (too) large? Then either the (input/output data-entry/measurement) instruments have to be refined or the system is probably of no practical use (independent of the function it *theoretically* realizes…). If we restrict ourselves, therefore, to systems with acceptable error margins, then the following three increasingly important shortcomings call for improvements:

First, notice that the range of data-entry/measurability is limited for every physical system (as already mentioned at the end of the previous section). From a practical point of view, there is not even a single physical system that realizes the addition function for all integers. The best that can be hoped for are systems that realize parts of that function. So, further restrictions have to be imposed on inputs and outputs: the domain of the abstract function has to be restricted to the interval $[x_{min},x_{max}]$ which is determined by 1) the data-entry/measurability constraints of devices producing the inputs and measuring the resulting outputs and 2) the range within which the physical system functions *normally* according to the laws of physics (a resistor, for example, does not behave *normally* when it starts to melt because the applied current is too high).

Second, the relation between input and output errors has hitherto been neglected. The only requirement imposed was that they be small enough to be of practical use. However, their relation becomes critical as soon as networks of circuits (which are constructed by connecting outputs of some circuits to inputs of others) are considered. Assume that three serially connected resistors form a circuit. Suppose that all resistors have an output error which is 10 times greater than their input error. Then the output error of the whole circuit will be 1000 times the input error (and that might already be

unacceptable). It seems, therefore, reasonable to require that the output error be less than or at most equal to the input error to allow for the construction of complex circuits, while at the same time keeping the overall error small.[20]

Finally, but most importantly, there are generally problematic cases in which the difference of two inputs to a system, $x_1$ and $x_2$, whose "error intervals" $[x_1-\varepsilon_{in},x_1+\varepsilon_{in}]$ and $[x_2-\varepsilon_{in},x_2+\varepsilon_{in}]$ overlap, is crucial. It is not clear at all how such a system could be of practical use. The main problem is that the function *f* which is supposed to be realized by *S* will be real-valued (as a consequence of *P* for most theories *P*), whereas the function describing the system's behavior (in terms of measurability) seems rather discrete-valued (a function from intervals of *Reals* to intervals of *Reals* with interval size $2*\varepsilon$). Or to use a metaphor: the abstract function is too "precise" for its "smudged" worldly counterpart. As long as there are overlapping regions between intervals in *F* which correspond to two distinct values of *f*, there will be cases in which, given a value *x*' in an overlapping region, one can neither determine the original *x* nor the resulting $f(x)$. The reason is that no bijection exists (hence no isomorphism either) between the *Reals* and disjoint intervals of the *Reals* of size $2*\varepsilon$ for any $\varepsilon>0$. Hence, it would make more sense to let *f* take values in the *Integers* (or *Rationals*) rather than the *Reals*. This would allow one to map discrete values to unique intervals if, in addition, the data-entry/measurement errors were small enough so that the intersection of any two images of discrete values, i.e., intervals, is empty. In such a system, all values of *f* could be measured/produced, hence the system would not only *theoretically*, but also *practically* (and verifiably) realize the function *f* (of course, only within the boundaries of $[x_{min},x_{max}]$). Notice that this last requirement implies that only certain *finite* ("timeless") functions can ever be totally realized (since there are smallest and largest values determined by the errors and/or physical possibilities of data-entry/measurement such as energy constraints, uncertainty, observability, etc. which, in turn, are determined by size constraints of the physical system…).

Taking all three modifications into account, we can define what it means to realize a part of a discrete-valued function *f* for given error terms $\varepsilon_{data}$ and $\varepsilon_{time}$ (which describe the measurement error of data and time, respectively—the output error is now assumed to be at most the input error, hence only one error term is needed):[21]

*Definition 7:* A "timeless" discrete function $f(x)$ with domain *D* and range *R* is (practically) realized within $[x_{min},x_{max}]\subseteq D$ realized by a system *S* (describable in a theory *P*) with errors $\varepsilon_{data}$ for input-output magnitudes and $\varepsilon_{time}$ for time if and only if the following conditions hold:

1. There exists a (syntactic) isomorphic mapping *I* from disjoint intervals of the "input domain" of *S* (where the interval length is $>2*\varepsilon_{data}$) to *D*
2. There exists a (syntactic) isomorphic mapping *O* from disjoint intervals of the "output domain" of *S* (where the interval length is $>2*\varepsilon_{data}$) to *R*
3. There exists a function *F* that describes the physical property (=behavior) of *S* for the given input-output properties over time (i.e., *F* is a mapping from the "input domain"×*RealTime* of *S* to its "output domain"×*RealTime* described in the language and by the laws of *P*) and a "delay-function" $d(x)$ from inputs to times (derived from

the $F$) such that for all $r \in RealTime$ and all $x \in [x_{min}, x_{max}]$ the following holds: $F(I^{-1}(x), r) = \langle O^{-1}(f(x)), r' \rangle$ (where $r' \in [r+d(x)-\varepsilon_{time}, r+d(x)+\varepsilon_{time}]$).[22]

This definition has achieved a great abstraction step: the magnitudes of physical dimensions have been "discretized" to guarantee practical applicability, i.e., continuity has been given up. However, the discrete values are still closely tied to the continuous values that describe the functionality of *S* in *P*. What is different, in more metaphorical terms, is that a grid (with box size $>2*\varepsilon_{data}$) has been superimposed on the functional graph, and only the points where the graph intersects with the grid are now considered.

Comparing definition 7 to definition 3, it becomes obvious how by incorporating physical and practical constraints, one arrives at a very restricted definition of what it means for a physical system to realize a function. Surely, we could have been satisfied with definition 4 or even definition 5, and for theoretical purposes they are just fine. But since we are interested in systems that we can actually *use* for various tasks (and might have to *build* in order to use them), we have to restrict ourselves to accessible ones, systems that can be utilized because they allow us to generate inputs and measure outputs.

One of the tasks that systems can be used for is "computation"—i.e., they can be used as "computers". For something to qualify as a "computer" it has to be at least a useable, physical system, which allows for data input and for measurable output, which works within reasonable time constraints and is sufficiently reliable.[23] Of course, other conditions will have be added depending on one's view of "computers" such as "executing an algorithm", etc. Some systems that realize functions according to definition 7 have all those properties by virtue of discretizing magnitudes and restricting the domain of possible values, others—where time plays a crucial role—require additional constraints on time (see definition 8). Not for all such systems will it be possible to define small error values, but some systems, those that are especially designed to facilitate the input/output-mappings for certain intervals, will be extremely reliable (because intervals are "far" apart and error values, for both time and magnitude, are small). Furthermore, these systems will be constructed to realize functions with a very small, finite domain (of mostly only *two* values!).

These properties together have often been summarized as the "digitality" of a system (see Haugeland, 1982, p. 215, or Haugeland, 1996, p. 9), i.e., the fact that there are "reliable" procedures for applying input and measuring output within the operating limits of the system and that there are finitely many distinct, discrete values given those limits. "Digital" means something like "of, relating to, or using *calculation* directly with *digits* rather than through measurable physical quantities" (Webster's dictionary, 1995, italics are mine).[24] Although systems that fall under definition 7 do not necessarily *calculate*, let alone with *digits*, they provide all the prerequisites that a system must have to *support* digits, since they *realize* discrete (timeless) functions by virtue of the physical laws that describe their behavior for a given set of inputs. Hence I will call them "digitality supporting systems".

Many electric components are especially designed to *support two digits*, so-called Boolean circuits (such as AND-gates, OR-gates, NOT-gates, etc.). Most of them realize

very simple functions such as the XOR function: $f(x,y)=0$ if $x=y$, otherwise $f(x,y)=1$ (where '0' and '1' are the two digits). Note that it is sometimes necessary to explicitly bring time into the picture again, especially to describe functions that use feedback (or to avoid unwanted behaviors in complex networks of Boolean circuits where the delay time of each unit becomes a critical factor in the overall performance). This requires us to make a final abstraction and change *f* in definition 7 from "timeless" to "discrete time" (analogous to definition 4), where the input and output domains of *f* now also contain "discrete times" from the set *Time* (normally modeled by integers):

*Definition 8:* A discrete function $f(x,t)$ with domain $D\times Time$ and range $R\times Time$ is (practically) realized within $[x_{min},x_{max}])\subseteq D$ by a system *S* (describable in a theory *P*) with errors $\varepsilon_{data}$ for input-output magnitudes and $\varepsilon_{time}$ for time if and only if the following conditions hold:

1. There exists a (syntactic) isomorphic mapping *I* from disjoint intervals of the "input domain" of *S* (where the interval length is $>2*\varepsilon_{data}$) to *D*
2. There exists a (syntactic) isomorphic mapping *O* from disjoint intervals of the "output domain" of *S* (where the interval length is $>2*\varepsilon_{data}$) to *R*
3. There exists a (syntactic) isomorphic mapping *T* from disjoint intervals of *RealTime* (where the interval length is $>2*\varepsilon_{time}$) to *Time*
4. There exists a function *F* that describes the physical property (=behavior) of *S* for the given input-output properties over time (i.e., *F* is a mapping from the "input domain"×*RealTime* of *S* to its "output domain"×*RealTime* described in the language and by the laws of *P*) such for all $t\in Time$ and all $x\in[x_{min},x_{max}]$ the following holds: if $F(I^{-1}(x),T^{-1}(t))=<X,Z>$, then $<X,Z>=<O^{-1}(x'),T^{-1}(t')>$ (where $f(x,t)=<x',t'>$).[25]

This definition looks very much like definition 4, except that all values are discrete instead of continuous. So, the above XOR function can now be captured as $f(x,y,t+1)=0$ if $x=y$ at *t*, otherwise $f(x,y,t+1)=1$ (and $f(x,y,0)=0$, say). And it can be used to define more complicated functions using feedback over time, such as the "oscillator" function *g* which alternates between 0s and 1s, once the input changes from 0 to 1: $g(x,t+1)=f(g(x,t),x,t+1)$ and $g(x,0)=0$ (it will be *realized* by a simple XOR gate where the output is fed back into its second input line).[26]

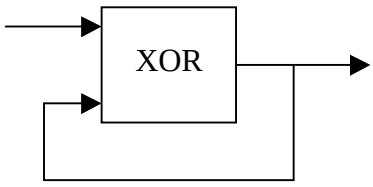


**Figure 4** The oscillator circuit: once the input is changed from 0 to 1, the output will oscillate between 0 and 1 as long as the input is 1.

To summarize the achievements so far: starting at the level of electric components and the physical descriptions of their properties, I defined the notion "a system realizes a

function" where the system consisted of components describable in terms of the physics of electricity. This notion was then—step by step—refined to account for practical problems regarding precision of production and measurement of signals, reliability, range of functioning, environmental influences, etc. The final definitions 7 and 8, respectively, related a discrete function (with a time parameter) to *digitality supporting systems*, i.e., systems that are of practical significance because inputs to them can be generated, and outputs (occurring after a limited time, possibly depending on the nature of the inputs) can be recognized and measured.[27]

## 10. From Circuits to Digital Systems

Definitions 7 and 8 have made two major abstractions (as compared to definitions 5 and 4), but neither times nor input/output magnitudes are completely "decoupled" from the concrete. The relation between input/output magnitudes as well as the one between points in time (i.e., the space-time metric) is still reflected in the input-output mappings together with their respective error terms. Values still stand for themselves and time points still have their unique place in the continuous flow of time, whereas in *genuine digital systems only* order matters, (spatial or value) distance and duration, i.e., metric distances, are secondary (if defined at all). It is not essential how "far apart" any two consecutive points are in time or space (of course, within limits) to be a particular digital system. In fact, different digital systems are shown identical *qua digital system* exactly by virtue of abstracting over physical peculiarities. Two steps must, therefore, be undertaken in order to prepare physical systems for *genuine digitality*: existential quantification over the particular error terms and a relaxation of the input-output mappings from isomorphisms to isomorphic embeddings.

The first takes care of spatial and temporal distance of value and time, respectively. Different physical systems will then realize, i.e., *be* the *same digital system*, even though they differ with respect to their error terms and the magnitudes of their input-output values. Take, for example, two physical systems (realizing binary AND-gates) which use different voltages for the binary values 0 and 1 and have different gate times. Both will now realize the *same AND-gate* (because of existential quantification over their error terms). To see that this is not sufficient, however, consider a ternary AND-gate, $AND_3$, using the values 0, 1, and 2 (the strong Kleene AND-function, say). Assume that two different physical systems, $S_3$ and $S_{10}$, are given. $S_3$ operates within [0,4] Volts and "maps" {0,1,2} to voltages according to $I(x)=O(x)=[x+1-0.1, x+1+0.1]$ with input-output and time errors of 0.1 (where the time mapping is given by $T(x)=[x+1-0.1, x+1+0.1]$). $S_{10}$ operates within [0,130] Volts and "maps" values from {0,…,9} to voltages according to $I(x)=O(x)=[(x+1)^2-0.5(x+1),(x+1)^2+0.5(x+1)]$ with input-output and time errors of $0.5(x+1)$ (the time mapping is again given by $T(x)=[x+1-0.1,x+1+0.1]$). Whereas the former is built to work for ternary systems only, the latter is thought to work for decimal systems. Notice that input-output mappings in $S_{10}$ are not linear, but quadratic (because they depend on the input-output error). Suppose now, that $S_{10}$ is used as $AND_3$ instead of $S_3$ (because the following voltages: "100" (±10) for "1", "10" (±10) for "0", and "50" (±10) for "2" are required in a given practical setting, say). Then $S_{10}$, realizing the

decimal AND-gate $AND_{10}$, also *realizes* $AND_3$ under the *isomorphic embeddings* $I^*$ and $O^*$ obtained by composing $I$ and $O$ with $E$ defined by $E(0)=1$, $E(1)=9$, and $E(2)=4$.[28] So, both systems realize the same (abstract) function, namely $AND_3$ under relaxed input-output constraints ("digits are decoupled from the grid imposed on $F$ by the input-output mappings and the error terms"). And the speed of the circuits is negligible as long as it stays within some limits determined pragmatically by the purpose of the circuit's use. Yet, it is important to keep in mind that these mappings are still not arbitrary (as they still preserve the relation between different inputs), rather they permit us to "pick and choose" specific values as digits (if only the distinctness of digits and neither their order nor relative magnitudes matters).

For some circuits, the current output depends only on the current input. For many others, the previous output will matter, too, especially for those circuits that allow (internal) feedback. The current output of the "oscillator" circuit defined in the previous section, for example, depends on the previous output and the current input. Hence, we arrive at functions that are realized by *digital systems* over time:

*Definition 9:* A discrete function $f(x,t)$ with domain $D\times Time$ and range $R\times Time$ ($D$ and $R$ finite) is realized by a system $S$ (describable in a theory $P$):

1. There exists a (syntactic) isomorphic embedding $I$ from disjoint intervals of the "input domain" of $S$ (where the interval length is $>2*\varepsilon_{data}$ for some $\varepsilon_{data}$ dependent on $S$) to $D$
2. There exists a (syntactic) isomorphic embedding $O$ from disjoint intervals of the "output domain" of $S$ (where the interval length is $>2*\varepsilon_{data}$ for some $\varepsilon_{data}$ dependent on $S$) to $R$
3. There exists a (syntactic) isomorphic mapping $T$ from disjoint intervals of *RealTime* (where the interval length is $>2*\varepsilon_{time}$ for some $\varepsilon_{time}$ dependent on $S$) to *Time*
4. There exists a function $F$ that describes the physical property (=behavior) of $S$ for the given input-output properties over time (i.e., $F$ is a mapping from the "input domain"$\times$*RealTime* of $S$ to its "output domain"$\times$*RealTime* described in the language and by the laws of $P$) such for all $t\in Time$ and all $x\in D$ the following holds: if $F(I^{-1}(x),T^{-1}(t))=\langle X,Z\rangle$, then $\langle X,Z\rangle=\langle O^{-1}(x'),T^{-1}(t')\rangle$ (where $f(x,t)=\langle x',t'\rangle$).[29]

Digital systems (according to definition 9) are physical systems that realize certain, discrete functions with finitely many input and output values (depending on their physical make-up). Only a finite set of discrete magnitudes corresponding to (some of the) input-output values together with the temporal order of applying input at a certain time and receiving output at some later time is retained (from the physical description of their functionality). The specifics (of *how* these finite values relate to each other or what the duration between any two points in time is) are ignored; hence the information about them is lost. It is given up in exchange for a purely theoretical treatment of these systems: *the physical (engineering) level of description is left in favor of a mathematical level of description*, where one can study functions realized by digital systems and their properties without recourse to the concrete using tools from mathematics and formal logic.

The system $S_{10}$ from above (which realizes $AND_3$ as well as $AND_{10}$) may elucidate this shift from the concrete into the abstract realm. Only three out of the ten values $S_{10}$ was designed for, found their use in $AND_3$, and their choice was merely guided by practical reasons; nothing theoretical forced it. In a sense, it was the set of all possible inputs (and outputs, i.e., the practical constraints) *together* with the function $E$ that determined whether the gate functioned as a ternary or as a decimal AND-gate. The input 1, for example, could be either mapped onto $I(1)=[3,5]$ or onto $I^*(1)=I(E(1))=I^*(9)=[95,105]$ Volts, depending on the input encoding that is used. In other words, the "embedding" $E$ suggests that $AND_3$ can be "realized" by $AND_{10}$, where this kind of "realizing" can be spelled out as "there exist embeddings $I_{3,10}$ and $O_{3,10}$ (in the above case $E$) between the domains and ranges of $AND_3$ and $AND_{10}$, respectively, such that for all $x$ and $y$ in the domain of $AND_3$ the following holds: $O_{3,10}^{-1}(AND_{10}(I_{3,10}(x),I_{3,10}(y)))=AND_3(x, y)$, so "comprising" would be a better term. The fact that $I^*$ is composed of $I$ and $E$ (i.e., $I^*=E\circ I$ for some $E$) shows *that and how* one can determine, *on purely mathematical grounds*, that $S_{10}$ realizes $AND_3$. It even provides a strategy to determine the class of functions that a physical system $S$ realizes under definition 9: take the function $f$ that $S$ realizes according to definition 8. Then all possible subsets of that function (up to renaming of the elements) will be realized by $S$ according to definition 9. Notice that instead of considering the relationship between a function and a physical system, the relationship between two functions becomes the focus of attention, as captured in the following definition:

*Definition 10:* A finite function $f(x)$ with domain $D_f$ and range $R_f$ is *comprised by* discrete, finite function $g(x)$ with domain $D_g$ and range $R_g$ if and only if the following conditions hold:

1. There exists a bijective mapping $I$ from $D_f$ into a subset of $D_g$
2. There exists a bijective mapping $O$ from $R_f$ into a subset of $R_g$
3. For all $x\in D_f$: $g(I(x))=O^{-1}(f(x))$.

Using definition 10, one can show that $S_{10}$ realizes every $AND_n$ for $n<=10$ by proving that every $AND_n$ for $n<=10$ is comprised by $AND_{10}$, and in consequence, that $S_{10}$ realizes the function $NOT_n(OR_n(NOT_n(x),NOT_n(y)))$ for all $n<=10$ (once its identity with $AND_n$ is proved for all $n$). The same technique can, furthermore, be used to prove that $S_{10}$ will *not* realize any $AND_n$ for $n>10$ (assuming the given physical description of $S_{10}$). And this kind of negative result could never be obtained within the physical theory itself!

## 11. Discussion

At this point I will interrupt the exposition, which could have continued to show how mathematical tools are applicable, how metatheoretical results can confine various classes of physical systems, how representations enter the picture, etc., but this is all part of another story. My goal in this paper was to show how one could bridge the gap between the concrete and the abstract, between physical systems and the computations they implement. It should be apparent now what the notional pair "computation-

implementation" had to make way for: the notion "realization of a function", i.e., what it means for a physical system *S* described in a theory *P* to realize a function *f*. Beginning with a definition of "realization of a function" that was closely tied to the theory describing the system, more and more constraints were incorporated leading to more and more restricted classes of physical systems (and thus restricted functions that are realized by those systems). Stepwise abstraction over physical dimensions and magnitudes eventually led to digitality, to the total abstraction over physical realizations. It enabled us to talk about functions such as AND-gates, registers, and even von Neuman computers, independent of their physical realization (while knowing at the same time, how these functions could eventually be realized physically in digital systems). Digital systems can, therefore, be viewed as *implementations* of the functions they realize, and these functions, in turn, will be the "abstract" *computations* that are realized by physical systems (i.e., digital systems). Note that even people who require that computations exhibit some kind of algorithmic structure should be satisfied, since digitality, i.e., the discreteness in space and time, lends itself naturally to algorithmic descriptions. In fact, every (reasonably-sized) finite state automaton, for example, can be described as a digital system (hence, implemented in a digital system described by some physical theory, e.g., circuit theory): just represent the FSA as a two-dimensional matrix ("lookup table"), where states are row indices, input characters column indices, and matrix elements contain the "next state" for the respective indices. This matrix, in turn, can be implemented using digital circuits (such as memories, counters, decoders, etc.).

The approach developed here differs significantly from the "state-to-state correspondence view": it does not require a notion of physical state, but determines directly the function which is realized by a physical system. Furthermore, it does not have to assume a particular computational formalism (to which the physical system exhibits a state-to-state correspondence), but can be related to computations described by any computational formalism (such as FSAs, TMs, PASCAL programs, cellular automata, etc.) *via* the function that these computations give rise to (see figure 5). It works for arbitrary levels of description of physical systems as well as different computational formalisms (as long as they provide a criterion of how to specify the function they compute). Thus, Putnam's construction does not pose a threat, since neither states/state types nor correspondence mappings have to be defined.

The main reason why the whole enterprise was guided almost exclusively by practical considerations is my focus on functions realized by systems that we can *recognize* and *use* (eventually as *computers*). Computing is an activity that can be recognized as such, initiated, performed, analyzed, etc.; an activity that is employed for various purposes. Searle's wall, for example, might realize a very complicated function at the level of fields, but we are not able (or at least not now) to utilize it for computation. That is not to say that only systems computing "by virtue of their digitality" are fit for computation. Various analog systems (and their physical properties) have been applied for fast, reliable computations.[30]

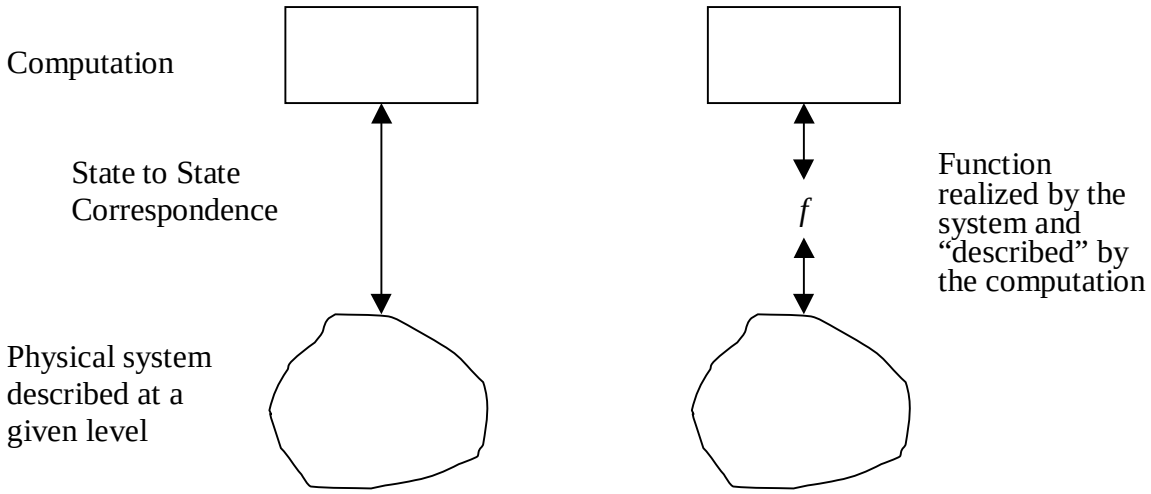


**Figure 5** The difference between the state-to-state correspondence view (left) and the "functional realization model" (right): in the latter, functions serve as mediators between physical systems and computations.

In the end, what kind of system is *useful* for computation is decided by *who is using it*: digital computers are built by us humans, because they are *useful* to us. Brains have evolved in animals and are obviously of use for them. Different kinds of currently not considered systems could be appropriate and applicable for computation. If "computation" is to mean "computing a function" and if this, in turn, is interpreted as "finding the value of the function for a given argument" (not necessarily "effectively"), then standard notions of computation are included as well as ones that do not rely on "manipulations of representations" (such as analog computations performed by VLSI chips, neural networks, etc.). Computational practice is certainly not limited to the former kind.

In a way, every system (described at some level *L*) that realizes a function could be seen as a computer, namely a computer computing that very function. But most of those "computers" are not useful for us; because we have no influence on their inputs and outputs, we might not be able to measure them or even recognize them as such. Hence, these systems do not qualify as "computers" (in a practical sense), although they could still be of interest for cognitive science. It might well be the case that cognitive systems are best described at level *L* and that at this level we cannot produce the right kinds of inputs and outputs, design and assemble the right kinds of components, etc. (because of technical problems). This result would be fatal for the artificial intelligence branch of cognitive science, since it would preclude the construction of *artificial* cognitive systems and very likely an understanding of cognitive systems in general). What makes us believe that this is not the case is that certain cognitive systems (e.g., human brains) can at some level be described as being digital (e.g., the symbol level). This abstraction over

the physical properties of human beings was essential to Turing's definition of the "ideal human computer" where human capacities to calculate were phrased in purely symbolic (i.e., digital) terms (token manipulation, rule application, etc., see also Haugeland, 1996). Whether the description of the brain at this or at a lower level (and the functions realized at that level) is essential to human cognition remains an empirical issue. It necessarily affects CCM, since the class of functions realized by digital systems is exactly what is commonly taken to be the extension of "computable". Assuming that the brain can only be described adequately at a level lower than the digital one, there are two possibilities: either the notion of computation has to be changed to the class of functions realized by systems described at that level, in which case CCM is true, otherwise CCM is false (if the "digital level of description" is adequate for the brain, however, then CCM is true). In any case, this is an empirical question that can only be decided by looking at the functions that (parts of) natural brains realize.

## 12. Conclusions

The driving force of this paper stems from the significance of the notions of computation and implementation for both cognitive science as well as the foundations of computer science. The lack of a tenable notion of implementation, however, renders CCM meaningless and shakes the fundaments of computer science. Since a general solution to the "implementation problem" in terms of a state-to-state correspondence theory of implementation seems extremely unlikely, a different approach is required, I claimed, one that starts with physical systems and their descriptions. The notions "computation" and "implementation" were given up in favor of various gradually refined notions of "realization of a function". These notions were developed from physical theories by abstracting over various physical properties culminating in the notion "digital system", which singles out systems that can be practically used and possibly even built. The issues at stake "do-ability", "feasibility", "usability", "practicability", "reliability", etc. are not only essential to computer science, but also of crucial importance to CCM: if natural cognitive systems do essentially exploit "errors", "unmeasurable (maybe infinitesimal) magnitudes", "quantum effects", etc., then they are *essentially non-digital systems* and cannot be described solely in terms of functions realized by digital systems (although some of their aspects might still be describable as being digital). However, this issue cannot be resolved *in principle*, but only by empirical investigations into the functionality of natural cognitive systems. The same empirical constraint is true with respect to the extension of the class of digital systems: the class of functions realized by digital systems depends crucially on (all possible) physical theories *P*. This is not a shortcoming, but a virtue of the approach. Whereas the kinds of physical systems that could be used for computing have to be determined theoretically (by looking at the restrictions posed by the various definitions), the class of physically possible systems depends on physical theories alone. I believe that the various notions of "realization of a function" provide a viable alternative to the standard notions of computation and implementation for this very reason. Furthermore, they open up a perspective which exposes the "computational" confinements that go hand in hand with the abstraction over physical peculiarities.

**Acknowledgements**
Special thanks are due to Brian Cantwell Smith for his valuable comments during the various phases of this paper. Also, I would like to thank an anonymous reviewer for helpful suggestions to improve readability.

## Notes

[1] It is important to notice the different implications of those attacks: whereas CCM *could* still be true for the former under a new, "revised" notion of computation, it will remain untenable for the latter unless a better notion of implementation is provided. As with almost every important and often used notion in computer science, "implementation", too, has a variety of different meanings which I discuss in Scheutz (1997). Here I will restrict myself to the reading suggested in the text, i.e., the relation between an abstract computation and a physical system *realizing* that computation.

[2] Since this paper is primarily concerned with the development of a positive account of implementation, I will restrict myself to a conceptual criticism of Putnam's arguments and objections to it. An analysis of Copeland's notion of implementation reducing it to a state-to-state correspondence view can be found in Scheutz (1998), a detailed study of various versions of the state-to-state correspondence view and their respective problems in Scheutz (1997).

[3] There are various extensions of Putnam's result that limit the range of possible responses: it can be proved for finite state machines with input and output, for Turing machines with only input, for state machines with (countably) infinitely many states, and, finally, for state machines that can read countably infinite strings of characters, see Scheutz (1997).

[4] At the level of fields, the question what exactly counts as an individual is a highly debated issue. It is not all that clear that Quantum Field Theory, for example, even contains individuation criteria for particles.

[5] Gödel (1958), for example, thought that human intuition, especially mathematical intuition, could exceed Turing computability, and his hunch is shared by influential logicians and scientists such as Feferman and Penrose.

[6] Notice that what is an *electric component* (e.g., resistors, capacitors, transistors, vacuum tubes, copper wires, etc.) is assured to be already determined and given. In particular, I will be talking about those electronic objects that I can buy in a store and weld into a board with copper connections on one or both sides to build things like radios, amplifiers, pocket calculators, and the like. Of course, a star has also a resistance (and, to some degree, resembles a resistor) and so does a molecule, but neither of them has the right size to be of use for a device that I can build, and this is all that matters if I want to listen to my favorite radio show or quickly calculate the accumulated interest in my bank account.

[7] Besides the fact that *all kinds of objects* can be described by the theory of "electric circuits", one could even doubt that it is clear what "object" means in this case. In other words, one could question the very notion of "object" and argue that circuit theory does not provide sufficient criteria for the individuation of its basic objects. Then the above level cannot be taken for granted, and one has to dig deeper into the metaphysical stuff to find substance and defining properties of objects (e.g., see Smith, 1996). From a pragmatical point of view, however, I believe that this step is not necessary: there are ways to find out if something is a (standard) "resistor", say, and even if a "thing" is not clearly a resistor, if it has the appropriate resistance (and that can be measured) and the appropriate shape, form, size, etc. it could be used as one.

[8] Actually, one reason to pursue this particular line of construction was exactly to avoid arguments about notions such as "counterfactual", "(natural) law", "law-likeness", "obeying a law", etc. A consent on what it means to "obey a physical law" or "to be a law of nature" is simply presumed.

[9] It seems to be a characteristic property of levels of description that if at a given level *n* a "what"-question can be answered, the corresponding "how"-question has to be answered at a lower level (if it can be answered at all).

[10] With "field axioms" I mean the set of axioms that describe properties of voltages and amperes with respect to "(voltage/current) addition" and "(voltage/current) multiplication". These axioms are necessary in order to define the "mathematical" properties of volts and amperes… Note that I left out quotation marks around all formulas in favor of readability.

[11] It is an interesting fact that we are completely used to talking about "quantitative physical properties" in terms of numbers, so used that it seems impossible to leave numbers out: all physical properties have a qualitative and a quantitative aspect (e.g., 100 kg or 20 m/sec). One likely reason is that the language of physics is built upon the language of mathematics, i.e., the language of (real/complex) analysis. There are, however, other ways of defining quantitative aspects in physics (e.g., nominalizing physics in the sense of Field).

[12] Cummins (1989) defines a notion called "a device satisfying a function", which *priama facie* looks very similar to my "a physical system realizes a function". There are two main differences, however: first, Cummins defines the relation of satisfaction only for functions that have the same input and output domain as the physical system, and second, he requires that input-output criteria for a given device can be specified which will determine if a given state of the system is an output value and if so, which state the corresponding input value was. His definition does not use a physical description of the device (which would eventually lend itself to a functional specification), but involves counterfactuals (to specify "state-transitions"), and at the end falls prey to Putnam's construction (since it does not specify the level of description of the physical device).

[13] One could "relax" the mapping by requiring that $D$ only be a subset of the input domain of $S$, thereby allowing the system to also realize functions that are "less complex" (in very much the same manner that the identity function over the *Reals* contains the one defined over the *Rationals* which in turn contains the one defined over the *Integers*). See section 10.

[14] It is worth pointing out that parameters differ from inputs in an essential way: they are normally adjusted until a desired value is reached (e.g., the volume of the amplifier before the guitarist starts to play), and then they remain set to this value, whereas the input will continue to change. This is, of course, only a rough cut, but it hints at the role of parameters in reconstructing the concept of "program" from physical peculiarities of certain systems.

[15] Adding adjustable parameters to physical systems actually marks a crucial step in my endeavor of *re*presenting the computational story. Once the transition from realizing one function to realizing multiple functions has been made, it is only natural to ask: "What *class of functions* does a system realize?". In the extreme case this class might turn out to be the whole class of functions itself (for a given definition of "realization of a function"). Those systems (the existence of which has to be argued for, of course) could then be called "universal" (with respect to the given definition).

[16] This idea could eventually give rise to a very different approach to computation, an approach that is *essentially* built upon the temporality of physical processes. Abstraction would, of course, be possible in many directions (e.g., towards digitality, see the next section), but duration and temporal order, being central, defining concepts, could never drop out during an abstraction process. This view on computation might come closer and do more justice to the behavior of what are called "embedded systems" (but I will not dwell on this here). Interestingly, operating or real-time system designers have been making a living out of coping with real-time constraints for quite some time.

[17] Note that if $d$ were totally unconstrained, very strange time lags would be allowed in principle. Fortunately this is not the case, since the physics of the system (of components in this case) will put restrictions on the time-dependencies between input and output (such that there will be no "jumps", no "points of discontinuity", etc., the function will rather be "smooth", "continuous", etc.): time dependencies will be law-like. More precisely, $d(x)$, being extracted from $F$, cannot be a "strange" function without rendering the whole system abstruse, if not absurd.

[18] Interestingly, the proof of Putnam's Realization Theorem relies on the so-called "Principle of Non-Cyclical Behavior", which is true of systems that are not shielded from environmental influences.

[19] Two notational remarks: I used '$f(X)$' (for a set $X$ and a function $f$) to mean '$\{f(x)|x\in X\}$', '<X,Y >' to denote the Cartesian product of $X$ and $Y$. Also, if time errors depend on inputs $x$, this can be accounted for by using error *function terms* (such as $\varepsilon_{tin}(x)$ and $\varepsilon_{tout}(x)$) instead.

[20] Note that there are really *two* output errors involved: the first is determined by the exactness of the measurement of the output—this is the one we have considered so far. The second is determined by the physical system itself, by the degree to which the system deviates from its formal description (e.g. imperfection and/or impurities of the material). Although the (overall) output error is really a combination of both individual errors, for theoretical purposes *one error term* that comprises both suffices (e.g., one could take the product of both error terms), especially, since it might not be clear which error actually contributes more/most to the overall error.

[21] The "construction" of the syntactic isomorphism is more complicated in this case, since intervals of a certain length need to be defined for input, output, and time domain. Then axioms for the discreteness of $f$ need to be added, etc. to allow for an appropriate formal treatment.

[22] One could require the weaker $F(I^{-1}(x),r) \subseteq <O^{-1}(f(x)),r'>$ to make the error resulting from an inaccurate description ("idealization") of the physical system itself explicit.

[23] Reliability is really a tricky issue. On the one hand, it is guaranteed by the laws of physics *qua* laws (i.e., that is exactly what it is to be a law: *to be reliable*), on the other hand, certain physical laws describe processes only up to probabilities. In those cases, "reliability" is automatically reduced to "(high) probability". In any case, how reliable a system will be at the end depends on the theory $P$ that describes its nature and functionality as well as on practical constraints (such as material, size, environmental influences, etc.).

[24] Haugeland (1982), p.214, argues that digits do not necessarily have to represent anything.

[25] Again, $<X,Z> \subseteq <O(x'),T(t')>$ would be the weaker requirement (see footnote 13).

[26] Notice that the functional definition of $g$ resembles the *scheme of recursion* where recursion is defined over time (this is no coincidence, but I will not be able to explicate the connection here).

[27] Note that all definitions assumed only one input domain $D$ and one output domain $R$, but, of course, they can be straight-forwardly extended to complex domains $D^n$ and $R^m$.

[28] Note that the order of the ternary elements induced by this mapping will be $0<2<1$, although nothing depends on the order in this case. However, there will be other cases where the order matters (e.g., in certain ADDER circuits).

[29] For those circuits whose output at $t+1$ only depends on their input at $t$, one can generally ignore the additional time place in the function and just consider finite one-place functions (e.g., the way AND-gates are normally defined), as long as it is understood that the output of the circuit occurs one "time step" after the input has been applied.

[30] It does not matter *how* a system realizes a function, i.e., if it computes using *representations* or if the result is obtained by the laws of physics (e.g., the distribution of charge in silicon), as long as the system is *digital from the outside*: a digital system can be treated as a "black box" with digital inputs and outputs, the inner organization does not matter for computation.

## References


Chalmers, D. J. (1996), 'Does a Rock Implement Every Finite-State Automaton?', *Synthese*, 108, pp. 310-333.

Chalmers, D. J. (1997), 'A computational Foundation for the Study of Cognition' (unpublished manuscript)

Copeland, B. J. (1996), 'What is Computation?', *Synthese*, 108, pp. 335-359.

Cummins, R. (1989), *Meaning and Mental Representation*, Cambridge, MA: MIT Press.

Endicott, R. P. (1996), 'Searle, Syntax, and Observer Relativity', *Canadian Journal of Philosophy*, v26, pp. 101-122.

Gandy, R. (1980), 'Church's Thesis and Principles for Mechanism', in *Proceedings of the Kleene Symposium* (J. Barwise, H. J. Keisler and K. Kunen, eds.), New York: North-Holland Publishing Company.

Gödel, K. (1958), 'Über eine bisher noch nicht benützte Erweiterung des finiten Standpunktes', *Dialectica*, 12, pp. 455-475.

Haugeland, J. (1982), 'Analog and Analog', *Mind, brain, and function*. Norman: University of Oklahoma Press.

Haugeland, J. (1996), 'What is Mind Design?' *Mind Design II*. Cambridge, Massachusetts: MIT Press.

Melnyk, A. (1996), 'Searle's Abstract Argument Against Strong AI', *Synthese*, 108, pp. 391-419.

Putnam, H. (1967), 'Psychological Predicates', reprinted as 'The Nature of Mental States', *The Nature of Mind*. D. Rosenthal (ed.) New York: Oxford University Press (1991).

Putnam, H. (1988), *Representation and Reality*, Cambridge: MIT Press.

Scheutz, M. (1997), 'Facets of implementation' (unpublished manuscript)

Scheutz, M. (1998), 'Do Walls Compute After All? – Challenging Copeland's Solution to Searle's Theorem Against Strong AI', in *Proceedings of the 9th Midwest AI and Cognitive Science Conference 1998*, AAAI Press, pp. 43-49.

Searle, J. (1992), *The Rediscovery of Mind*, Cambridge, Massachusetts: MIT Press.

Smith, B. C. (1996), *The Origin of Objects*, Cambridge, Massachusetts: MIT Press.

Smith, B. C. (1998), *The Age of Significance*. Vol 1. (forthcoming)

Turing, A. M. (1936), 'On Computable Numbers, with an Application to the Entscheidungsproblem'. *Proceedings of the London Mathematical Society*, *Series 2*, 42, pp. 230 –265.

Van Gelder, T. J. (1998), 'The Dynamical Hypothesis in Cognitive Science' (target article forthcoming in *The Behavioral and Brain Sciences*)