Chapter 2
Computation in Physical Systems: A Normative Mapping Account

Paul Schweizer

Abstract The relationship between abstract formal procedures and the activities of actual physical systems has proved to be surprisingly subtle and controversial, and there are a number of competing accounts of when a physical system can be properly said to implement a mathematical formalism and hence perform a computation. I defend an account wherein computational descriptions of physical systems are high-level normative interpretations motivated by our pragmatic concerns. Furthermore, the criteria of utility and success vary according to our diverse purposes and pragmatic goals. Hence there is no independent or uniform fact to the matter, and I advance the 'anti-realist' conclusion that computational descriptions of physical systems are not founded upon deep ontological distinctions, but rather upon interest-relative human conventions. Hence physical computation is a 'conventional' rather than a 'natural' kind.

Keywords Computational Theory of Mind · Physical computation · Simple mapping account · Pancomputationalism · Computational stance

2.1 Introduction

What is computation? There are two basic ways to look at the issue: (1) in theory, as a type of *mathematical 'process'* – as something that belongs to a purely abstract and formal domain, like topology, set theory or real analysis; and (2) in practice, as the activity of certain *physical systems* – as what computers do, where a *computer* is a concrete device that exists in actual space and time. The connection between these two perspectives is generally conceived to lie in the *implementation* relation: a physical system or device performs a computation when

P. Schweizer (✉)
Institute for Language, Cognition and Computation, School of Informatics, University of Edinburgh, Edinburgh, UK
e-mail: paul@inf.ed.ac.uk

D. Berkich, M. V. d'Alfonso (eds.), *On the Cognitive, Ethical, and Scientific Dimensions of Artificial Intelligence*, Philosophical Studies Series 134,
https://doi.org/10.1007/978-3-030-01800-9_2

it ‘implements’ or ‘realizes’ a particular abstract formalism. However, specifying the criteria under which the implementation relation properly obtains has proved surprisingly subtle and controversial, and there are a number of opposing views on the constraints that must be satisfied in order for a physical system to count as a ‘genuine’ implementation.

2.2 A Simple Mapping Account

A very straightforward and elegant account articulated by Putnam (1988) and others is based on a simple mapping between physical structure and abstract formalism. Accordingly, a physical system *P* performs a computation *C* just in case there is a *mapping* from the actual physical states of *P* to the abstract computational states of *C*, such that the transitions between physical states reflect the abstract state transitions as specified by the mapping. The minimalism, neutrality and generality of the Simple Mapping Account (henceforth SMA, adopting the terminology of Piccinini 2015a) make it a natural choice as the in-principle standard for physical implementation – it takes the Mathematical Theory of Computation (MTC) as its starting point and adds no substantial assumptions. And because it adds no further assumptions or restrictions, SMA is in an important sense maximally liberal – there will exist abstract mappings from a *huge* class of physical systems and processes to an equally huge class of computational formalisms.

And there is a clear sense in which this is a significant theoretical virtue. It is standard practice in computer science to apply computational descriptions to various physical systems *at will*, simply on the condition that the mapping yields an interesting or useful perspective. For example, simple physical devices such as parking ticket dispensers or traffic light controllers can be modelled in terms of Finite State Machines, without any reference to the original intentions of their designers nor to the actual details of their internal causal structure. In such cases, computational ascriptions constitute an *idealized* depiction, one that abstracts away from many actual features of the device to yield a simplified *formal model* of selected aspects of the device. This is quite analogous to applying mathematical formalisms such as differential equations to various physical systems to characterize aspects of their trajectories through state space. In both cases, the mapping from physical phenomena to mathematical formalism is highly reliant on both idealization and approximation, and deliberately neglects many aspects of the internal causal mechanisms.

In this type of endeavor, *which* aspects of the system are selected for abstract modelling is not fundamental to the system *per se*, but instead remains a question of human choice relative to our interests and goals. There are any number of different perspectives and levels for describing the very same system, and none of them is privileged. A traditional spring-driven analogue clock can be formally modelled at various *microphysical* levels – at a subatomic level in terms of quantum mechanical

processes and interactions, and at a higher microphysical level in terms of molecular thermodynamics. In the latter case, it could also be described in more abstract functional terms as a temperature detector, where the mean molecular kinetic energy of its metallic components tracks the ambient atmospheric temperature. And it can be described and modelled at various *macrophysical* levels as well, such as an intricate classical mechanism with states evolving in accord with continuous real valued equations. It could also be described in more idealized *conventional* terms, where certain selected continuous features are broken into discrete segments and given a chronological interpretation. And yet again, this relatively advanced design level stance could be ignored, and the object could be given a more rudimentary functional depiction, e.g. where its size and inertial properties make it useful as a doorstop.

For computation to remain an unfettered, and maximally adaptable mathematical tool, like set theory or topology, it is requisite that no fixed or preconceived limits be placed upon its potential range of physical interpretation. And indeed, SMA exemplifies this neutrality and universality with respect to the possible relations between abstract formal structure and 'concrete' physical phenomena. In this vein, Putnam (1988) gives a technical proof of the theorem that every open physical system implements every (inputless) Finite State Machine (FSM). He provides a generic depiction of a physical system as a bounded, continuous region of space-time, and the basic idea is that the region is held constant but sliced up in an as many different ways as one likes in order to define a sequence of disjunctive 'physical states' that can be mapped to any given run of a FSM.

And Searle famously promulgates the universality of SMA with the claim that virtually *any* physical system can be interpreted as implementing virtually *any* formal procedure. For example, Searle (1990) asserts that the molecules in his wall could be interpreted as running the WordStar program. The claim is simply put forward with no further defense, but Copeland (1996) provides a proof of what he calls 'Searle's Theorem', which he observes is essentially a notational variant of Newman's (1928) objection to Russell (although Copeland then goes on to reject SMA).

This broad-minded position on physical computation arises as the natural inverse of the standard and uncontroversial view that abstract formal procedures, as such, are *multiply realizable*. It's clearly possible to implement the *very same* computational formalism using vastly different arrangements of mass/energy. Following notation and terminology introduced in Schweizer (2012), let us call this top-down feature 'downward multiple realizability', wherein, for any given formal procedure, this *same* abstract formalism can be implemented via an arbitrarily large number of *distinct types* of physical systems. And let us denote this type of downward multiple realizability as '↓MR'. The basic perspective advocated by Putnam and Searle then goes in the reverse direction. Let us call the bottom-up view that any given *physical system* can be interpreted as implementing an indeterminately large number of different *computational formalisms* 'upward MR' and denote it as '↑MR'. The basic import of ↑MR is the non-uniqueness of computational

ascriptions to particular configurations of mass/energy. In the extreme versions of ↑MR propounded by Putnam and Searle, it is not simply a case of non-uniqueness, but rather there are apparently no significant constraints at all – it is held to be possible to interpret virtually any open physical system as realizing virtually every computational procedure. Let us call this more extreme version '*universal* upward MR' and denote it as '↑MR*'. ↑MR* is noteworthy in that it provides the theoretical limit case in terms of abstraction away from physical specifics or limitations, and in this sense is comparable to the idea that, e.g., any physical object can be an element in a limitless number of distinct sets.

2.3 The Computational Stance

Many philosophers have found the degree of liberality induced by SMA objectionable. Historically, these objections stem from the conflict between critics *versus* proponents of the Computational Theory of Mind (CTM). Critics of CTM have used SMA to argue that a computational approach to the mind is empirically vacuous. These 'trivialization' arguments hold that, *a la* ↑MR*, a mapping will obtain between virtually any physical system and virtually any formalism, which in turn is construed as fatally undermining CTM, since whatever computational procedures are held to account for our cognitive attributes will also be realized by a myriad of other 'deviant' physical systems, such as buckets of water and possibly even stones. Hence by CTM it would seem to follow that such obviously insentient systems have the same cognitive attributes that we do, which is then taken as a *reductio ad absurdum* disproof of CTM.

In response to ↑MR* and the associated trivialization claims, a host of authors, including Fodor (1981), Maudlin (1989), Chrisley (1994), Chalmers (1996), Copeland (1996), Shagrir (2001), Block (2002), Sprevak (2010), Milkowski (2013), Rescorla (2014), Piccinini (2015b) advocate additional constraints on the implementation relation, so that it is no longer a 'simple' or theoretically neutral mapping. In effect, these restrictions serve to preclude a vast number of physical systems from the domain of the mapping function, in an attempt to separate 'true' or 'genuine' implementations from the many presumably 'false' cases countenanced by SMA. These constraints include: causal, counterfactual, semantic, and mechanistic/functional criteria.

However, I advance quite a different type of response to the situation. First, instead of attempting to 'save' CTM by constraining the account of physical implementation, I hold that SMA does not actually constitute a threat to scientifically plausible versions of CTM. No one thinks that SMA 'threatens' electrical engineering or our ability to design and utilize sophisticated computational artifacts, and in my view, the particular version of CTM that *is* undermined by SMA is not one that should be accepted in any case. Second, I argue that none of the proposed constraints provides a truly general and satisfactory 'realist' account of

physical implementation – indeed, none succeeds at providing a globally applicable necessary condition. So I advocate retaining a very liberal SMA view of physical implementation, that derives from the basic insights of Turing, Kripke, Putnam and Searle, while rejecting the standard anti-CTM conclusion of the trivialization arguments.

In line with SMA and ↑MR, I argue that computation is not an 'intrinsic' property of physical systems, in the sense that (a) it is founded on an observer-dependent act of ascription, upon an entirely *conventional* correlation between physical structure and abstract formalism. Furthermore, (b) this conventional mapping is essentially prescriptive in nature, and hence projects an outside *normative* standard onto the activities of a purely physical device. In this manner we adopt what could be termed a 'Computational Stance' towards physical systems. This approach is in some ways comparable to Dennett's (1981) Intentional Stance, wherein intentional states such as beliefs and desires are not posited as objectively real phenomena. Instead, they are treated as mere 'calculational devices' or '*abstracta*' in Reichenbach's sense (like point masses and perfectly frictionless surfaces in classical mechanics), used to predict observable events but without any additional ontological commitments.

Analogously, I would construe abstract computational states on a similar footing. In the case of our purpose-built artifacts, these abstract states are *idealized* formal notions that we employ to describe such devices from a higher design-level perspective. Classic digital computation is rule-governed syntax manipulation, and as such is no more intrinsic to physical configurations than is syntax itself. Furthermore, discrete states are themselves idealizations, since the physical processes that we interpret as performing digital computations are continuous (in the standard non-quantum case). Thus discrete states do not literally correspond to the underlying causal substrate. We must *abstract away* from the continuity of actual physical processes and impose a scheme of conventional demarcations to attain values that we can then *interpret* as discrete. Hence this elemental building block of digital procedures must be projected onto the natural order from the very beginning (as Turing observed in 1950), and in this respect is a convenient fiction rather than a literal depiction.

Dennett holds that there is no internal matter of fact distinguishing systems that 'really' possess intentional states from those which do not – the strategy only requires us to view the system *as if* it possessed such states. Hence there is nothing in principle to stop one from depicting a stone as an intentional system if one so chooses. In a similar vein, I would argue that there is no deep or metaphysically grounded fact regarding whether or not a physical system 'really' implements a given computational formalism. In the case of artifacts such as my desk top computer, I can gain a huge increase in the ability to predict its future states if I adopt a computational stance as opposed to viewing it as a brute physical mechanism. And this is because it has been designed and constructed for exactly this purpose, just as an electric toaster has been designed and constructed to perform a particular function. In contrast, a stone has not been so designed, and the pragmatic value of viewing it in computational terms will be rather limited.

2.4 Critique of the Causal Account

I will now critically address some of the proposed constraints on SMA, with the aim of showing that none provides a principled necessary condition for physical computation. The causal account supplies one of the most natural and intuitively compelling constraints on the implementation relation. Chalmers (1996), for example, contends that it is a necessary condition that the pattern of abstract state transitions must be the image under the mapping of an appropriate transition of physical states of the machine, where the relation between succeeding physical states in this sequence is governed by *proper causal regularities*. Furthermore, these regularities are supposed to 'mirror' the structure of the abstract formalism. The imposition of such a constraint will screen off a vast number of Putnam's sequences of physical states, with the aim of reducing the domain of the mapping function to a tiny subset of purportedly 'legitimate' cases of implementation.

However, I argue that the causal constraint is too stringent in general and rules out a significant number of cases which should not be excluded. And although it's a more specialized and sophisticated approach, the mechanistic/functional account shares some key features with the causal, so many of the following criticisms carry over to this account as well. A basic problem with causal and mechanistic approaches is that they place emphasis on the wrong level of conceptual analysis. Rather than addressing the question of *whether or not* a given configuration of mass/energy implements a given computational formalism, causal considerations instead address the lower level and divergent practical question of *how*, in certain circumstances and over limited spans of time, this implementational sequence is mechanically generated.

The inessential status of causal structure can be elucidated with the observation that the key factor in judging *that* a given configuration of mass/energy implements a particular computational formalism is *simply because*, according to our abstract blueprint, the *correct* series of physical state transitions actually occurs. As an exemplary case of where appeal to causal regularities is completely irrelevant to determining whether or not a given sequence of states counts as an implementation, consider Turing's original (1936) heuristics, where the paradigm of actualized computation is a *human computor*, meticulously following a program of instructions and executing computations by hand with pencil and paper. In this seminal and classic example of concrete realization, the transitions from one state to the next are not governed by causal regularities in any straightforward mechanical sense. When I take a table of instructions specifying a particular abstract TM and perform a computation on some input by sketching the configuration of the tape and read/write head at each step in the sequence, the transitions sketched on the paper are *not themselves* causally connected: as in the virtual machine states in standard computers, one sketch in the sequence in no way causes the next to occur.

In terms of the 'causal' factors underpinning their occurrence, it is primarily through my understanding of the instructions and intentional choice to execute the procedure that the next stage in the sequence appears. But my complex behaviour as a human agent deliberately following instructions is not something that we currently have any hope of being able to recast in terms of causal regularities at the purely physical level of description. Furthermore, whatever causal factors at this level *do* ultimately underwrite my ability to execute the procedure, they will be exceedingly convoluted and indirect, and there is no reason to believe that they will 'mirror' (or even remotely resemble) the structure of the formalism. In cases of *intentionally mediated* causation, we accept the sequence of configurations on the paper as an implementation of the program, not because we have the faintest idea of the underlying causal story, but rather because the sequence itself is *correct* and can be seen to follow the procedural rules. In other words, the projected mapping, *a la* SMA, has been preserved.

To continue the example, consider the following 3 state Turing machine ***M*** given by the four quadruples:

$$q_1 1 R q_1 \quad q_1 0 1 q_2 \quad q_2 1 L q_2 \quad q_2 0 R q_3$$

The first element in each quadruple (e.g. q_1 in the first case) is the current state, the second element is the currently scanned symbol (either 1 or 0) the third element is the overt action (*move* R or L one square, or *print* a 1 or a 0), and the last element is the covert 'act' of entering the next state. Now suppose I'm confronted with an initial tape configuration

$$01100\ldots \quad \text{(all other squares to the right are blank)}\ .$$

Armed with the foregoing explication of the quadruple notation, along with a few basic operational conventions (as described in Boolos and Jeffrey 1989), I can act as a perfectly good human computor and manifest a physical implementation of the respective Turing machine computation. With pencil and paper I can perform the sequence of 6 transitions determined by the input configuration and then halt. Indeed, I've now keyed into the digital file generating this document the very same sequence that I sketched in my notebook, and have thus produced an alternate physical realization of the same computation. The machine starts in its lowest numbered state reading the leftmost non-blank square (where the contents of a square are indicated by the corresponding digit in the tape string). An underline indicates the currently scanned square, and the number below this indicates the current state. The machine halts when it enters a state for which there is no instruction.

$q_1 1 R q_1$; $q_1 0 1 q_2$; $q_2 1 L q_2$; $q_2 0 R q_3$ on input 01100…

Start 0<u>1</u>100…
1
01<u>1</u>00…
1

011<u>0</u>0…
1
011<u>1</u>0…
2
01<u>1</u>10…
2
0<u>1</u>110…
2
<u>0</u>1110…
2
0<u>1</u>110… Halt
3

It's important to note that the foregoing sequence of configurations is not just a *linguistic description of* a possible physical implementation. Instead, the actual syntactic tokens are *themselves* concrete realizations extended in physical space-time. Manipulating syntactic tokens on a piece of paper *is* a transformation of the physical environment that itself constitutes a realization of the abstract formalism. And the same is true of the sequence of symbols generated above – it's a physical implementation of the abstract TM computation generated by Microsoft Word.

But what is the *causal structure* underling the Microsoft implementation? It doesn't really matter. The entries in this sequence bear no decipherable *causal* relations to each other – they're simply generated by what is stored in the digital file that is stored in the computer connected to the monitor. The actual computation in space-time appearing as I type is a sequence of illuminated patterns projected onto the screen, not supported by any causal regularities that 'mirror' the structure of the Turing machine program. It's surely true that every event must have a cause, but my point is that *surface inspection alone* reveals that this sequence is a proper realization of the specified TM program on input 01100 . . . To arrive at the judgement, we do not need to know *anything* about the causal mechanisms whereby this sequence was produced.

And what is the *semantic interpretation* of the Microsoft implementation? Again, it doesn't really matter. The computation itself is comprised of rule governed syntactic transformations. How these transformations are then semantically construed is superfluous to the execution of the program. If we choose, we *can* interpret $\boldsymbol{M}$'s activity as computing the function $f(x) = x + 1$ on positive integers expressed in monadic notation (and which halts on the same square at which it starts), so that the foregoing sequence of configurations is a computation of $f(2) = 2 + 1 = 3$. However, this is clearly not essential to the formal procedure itself.

And what *would have happened if* a different input string had been attempted? Again, it doesn't really matter. What matters is that, in accord with the formal procedure, the foregoing sequence is *correct* – it satisfies the essential normative specification as a series of rule governed transformations on the input specified.

2.5 Implementation as Proof in First-Order Logic

Each quadruple in the TM program can be seen as a conditional instruction, so that, e.g., the first quadruple is the conditional: *if* in state q_1 reading a 1, *then* print a 0 and enter state q_1. Hence it is the *logical* form of the if-then statement that captures the significance of the TM instruction, and this is all that must be satisfied by an implementation. Again, this is a quintessentially *normative* constraint, and it's a basic fact of logic that the truth-functional character of the material conditional does not imply *any causal connection* between antecedent and consequent.

This same fundamental point is made even more graphic by noting that Turing machine computations can be formalized in first-order logic with identity (FOL=). Each quadruple instruction can be rendered as a *universally quantified conditional* indicating the result of executing the instruction. In providing the details of the formalization, our object language $\boldsymbol{L}$ for FOL = will contain the symbols $\mathbf{o}$ and $'$ as distinguished vocabulary items, where $\mathbf{o}$ is a singular constant that, under the intended interpretation $\mathcal{I}$, denotes the number 0, and where $'$ is a 1-place function symbol which under $\mathcal{I}$ denotes the successor function. With these resources we can construct canonical numerals intended to denote numbers in the obvious fashion, e.g., where $\mathbf{o}'$ is the numeral for the number 1, $\mathbf{o}''$ the numeral for 2, etc.

In order to formalize the very simple machine $\boldsymbol{M}$ depicted above, we can make do with the assumption that the operant squares are unbounded only to the right. Furthermore, a blank square is construed as containing the symbol '0', and only finitely many squares are ever non-blank (i.e. contain the symbol '1'). To begin the formalization, let all the operant squares of the tape be labelled by a natural number, with the leftmost such square labelled with 0, the next with 1, etc. (the labelling number is distinct from the symbol occurring in the square). We adopt the convention that the positive integer input is expressed in monadic notation, with the leftmost '1' occurring in square number 1. At the start of the computation, all non-input squares of the tape are blank, and the machine starts in state 1 reading square 1.

Let t be the 'time' variable ranging over steps in the computation. We need two final FOL vocabulary items: for each state q_i of a given machine, pick a 2-place predicate $\mathbf{Q}_i$. For each symbol S_j the machine can read/write, pick a 2-place predicate $\mathbf{S}_j$ (in this case there are only two). The domian $\mathcal{D}$ of the intended interpretation $\mathcal{I}$ is the set of natural numbers, and $t\mathbf{Q}_i x$ is true in $\mathcal{I}$ iff at time t $\boldsymbol{M}$ is in state q_i scanning square number x, and $t\mathbf{S}_j x$ is true in $\mathcal{I}$ iff at time t the symbol S_j is in square number x. With these details in place we can now proceed to formalize $\boldsymbol{M}$'s program of instructions.

The first quadruple $q_1 1 R q_1$ is rendered as the 'axiom' **A1**

$$\forall t \forall x \forall y \left[(t \mathrm{Q}_1 x \wedge t \mathrm{S}_1 x) \rightarrow \left(t' \mathrm{Q}_1 x' \wedge \left(t \mathrm{S}_0 y \rightarrow t' \mathrm{S}_0 y \wedge t \mathrm{S}_1 y \rightarrow t' \mathrm{S}_1 y \right) \right) \right]$$

Under the intended interpretation this axiom 'says that' *if* machine ***M*** is in state q_1 at time t scanning square number x on which the symbol S_1 (= 1) occurs, *then* at time $t + 1$ ***M*** is in state q_1 scanning square $x + 1$, and in all squares the same symbol appears at time $t + 1$as at time t.

Various authors (including Chalmers 1996 and Copeland 1996) have objected to Putnam's proof because it relies on material conditionals, and it is claimed that more powerful *counterfactual* machinery is required to account for possibilities other than the input actually given. However, it is significant to note that the above universally quantified conditional ranges over *all times* and *all squares* in *any* computation, and hence exhaustively covers all relevant possibilities.

$q_1 0 1 q_2$ is rendered as **A2**

$$\begin{aligned} &\forall t \forall x \forall y \left[(t \mathrm{Q}_1 x \wedge t \mathrm{S}_0 x) \right. \\ &\quad \left. \rightarrow \left(t' \mathrm{Q}_2 x \wedge t' \mathrm{S}_1 x \wedge \left(\mathrm{y} \neq \mathrm{x} \rightarrow \left(t \mathrm{S}_0 y \rightarrow t' \mathrm{S}_0 y \wedge t \mathrm{S}_1 y \rightarrow t' \mathrm{S}_1 y \right) \right) \right) \right] \end{aligned}$$

$q_2 1 L q_2$ yields **A3**

$$\forall t \forall x \forall y \left[\left(t \mathbf{Q}_2 x' \wedge t \mathbf{S}_1 x' \right) \rightarrow \left(t' \mathbf{Q}_2 x \wedge \left(t \mathbf{S}_0 y \rightarrow t' \mathbf{S}_0 y \wedge t \mathbf{S}_1 y \rightarrow t' \mathbf{S}_1 y \right) \right) \right]$$

$q_2 0 R q_3$ yields **A4**

$$\forall t \forall x \forall y \left[(t \mathrm{Q}_2 x \wedge t \mathrm{S}_0 x) \rightarrow \left(t' \mathrm{Q}_3 x' \wedge \left(t \mathrm{S}_0 y \rightarrow t' \mathrm{S}_0 y \wedge t \mathrm{S}_1 y \rightarrow t' \mathrm{S}_1 y \right) \right) \right]$$

The set {**A1**,**A2**,**A3**,**A4**} **formalizes** ***M***'s program.

Next two arithmetical axioms are needed to govern the behavior of $'$ and **<**. The first axiom says that each integer is the successor of exactly one integer: $\mathbf{A}'$

$$\forall z \exists x \left(z = x' \right) \wedge \forall z \forall x \forall y \ \left(\left(z = x' \wedge z = y' \right) \rightarrow x = y \right).$$

The axiom governing **<** states that: **A<**

$$\begin{aligned} &\forall x \forall y \forall z \ (x < y \wedge y < z \rightarrow x < z) \wedge \forall x \forall y \left(x' = y \rightarrow x < y \right) \\ &\quad \wedge \forall x \forall y \, (x < y \rightarrow x \neq y) \ \text{(needed for the entailment relation below)} \end{aligned}$$

Finally, for the initial configuration with '01100' as starting input

$$(t = 0 \text{ in state } \mathrm{q}_1 \text{ reading square 1}) : \mathbf{A0}$$

$$\mathbf{o}\mathbf{Q}_1\mathbf{o}' \wedge \mathbf{o}\mathbf{S}_1\mathbf{o}' \wedge \mathbf{o}\mathbf{S}_1\mathbf{o}'' \wedge \forall y \ \big((y{\neq}\mathbf{o}' \wedge y{\neq}\mathbf{o}'') \rightarrow \mathbf{o}\mathbf{S}_0 y\big)$$

Let $\Delta = \{\mathbf{A1},\mathbf{A2},\mathbf{A3},\mathbf{A4}, \mathbf{A}',\mathbf{A}<,\mathbf{A0}\}$

Now Δ completely formalizes the 'actions' of machine ***M*** on input '01100', and each step n in the previously sketched sequence of configurations, constituting the computation on input '01100', is s*yntactically encoded* by a sentence $\mathbf{T}_n$ in FOL=. Furthermore, the sentence $\mathbf{T}_n$ is ***logically entailed*** by Δ.

For $t = 1$ the sentence $\mathbf{T}_1$:

$$\mathbf{o}'\mathbf{Q}_1\mathbf{o}'' \wedge \mathbf{o}'\mathbf{S}_1\mathbf{o}' \wedge \mathbf{o}'\mathbf{S}_1\mathbf{o}'' \wedge \forall y \ \big((y{\neq}\mathbf{o}' \wedge y{\neq}\mathbf{o}'') \rightarrow \mathbf{o}'\mathbf{S}_0 y\big)$$

For $t = 2$ the sentence $\mathbf{T}_2$:

$$\mathbf{o}''\mathbf{Q}_1\mathbf{o}''' \wedge \mathbf{o}''\mathbf{S}_1\mathbf{o}' \wedge \mathbf{o}''\mathbf{S}_1\mathbf{o}'' \wedge \mathbf{o}''\mathbf{S}_0\mathbf{o}''' \wedge \forall y \ \big((y{\neq}\mathbf{o}' \wedge y{\neq}\mathbf{o}'' \wedge y{\neq}\mathbf{o}''') \rightarrow \mathbf{o}''\mathbf{S}_0 y\big)$$

For $t = 3$ the sentence $\mathbf{T}_3$:

$$\mathbf{o}'''\mathbf{Q}_2\mathbf{o}''' \wedge \mathbf{o}'''\mathbf{S}_1\mathbf{o}' \wedge \mathbf{o}'''\mathbf{S}_1\mathbf{o}'' \wedge \mathbf{o}'''\mathbf{S}_1\mathbf{o}''' \wedge \forall y\big((y{\neq}\mathbf{o}' \wedge y{\neq}\mathbf{o}'' \wedge y{\neq}\mathbf{o}''') \rightarrow \mathbf{o}'''\mathbf{S}_0 y\big)$$

$$\vdots$$

For $t = 7$ the sentence $\mathbf{T}_7$:

$$\mathbf{o}''''''\mathbf{Q}_3\mathbf{o}' \wedge \mathbf{o}''''''\mathbf{S}_1\mathbf{o}' \wedge \mathbf{o}''''''\mathbf{S}_1\mathbf{o}'' \wedge \mathbf{o}''''''\mathbf{S}_1\mathbf{o}'''$$
$$\wedge \forall y \ \big((y{\neq}\mathbf{o}' \wedge y{\neq}\mathbf{o}'' \wedge y{\neq}\mathbf{o}''') \rightarrow \mathbf{o}''''''\mathbf{S}_0 y\big)$$

M has no instructions for q_3 and hence will halt if it enters this state. So the 'canonical' Halting Sentence **H** for this machine is

$$\exists t \exists x \ (t\mathbf{Q}_3 x \wedge t\mathbf{S}_0 x) \vee \exists t \exists x \ (t\mathbf{Q}_3 x \wedge t\mathbf{S}_1 x)$$

and it's provable (by mathematical induction) that $\Delta \vDash \mathbf{H}$, since $\Delta \vDash$ T8 and T8 $\vDash$ **H**.

Logical entailment is an abstract mathematical relation, but a *particular proof* is a concrete syntactic phenomenon extended in physical space-time. In this manner, the foregoing Turing machine computation is equivalent to a proof in FOL=, and any such proof carried out with pencil and paper, following the rules of your favorite first-order deductive system, counts as a *physical implementation* of the computation.

It seems a very strange and implausible view to maintain that the property of being a proof in first-order logic is constrained by underlying causal regularities or mechanistic features. Indeed, when I mark student exams in my *Introduction to Logic* course, considerations of underlying causal regularities and biological mechanisms play no role whatever in determining whether some sequence of formulas is or is not a proof. The only thing that matters is whether or not the rules have been correctly followed, and this is a purely normative consideration. And since a proof of the relevant sort counts as an implementation of a Turing machine computation, it follows that causal regularities likewise have no bearing on the status of such implementations. Indeed, part of the reason that underlying causal considerations are the wrong level of analysis is that there is no sense in which error or malfunction can occur when viewed from this basic physical perspective. This thread will be resumed in Sect. 2.7.

The foregoing *counterexamples* show that causal and mechanistic factors do not impose a necessary condition on physical implementation. Instead, the *only* necessary condition is that the intended mapping, *a la* SMA, is preserved. In particular, we don't need to take into account the mechanics of *how* this success has been achieved in order to judge *that* it has it has occurred. And indeed, this is directly comparable to other abstract, rule governed activities such as chess. A game of chess is constituted by a sequence of moves on a geometrically defined board. Like computations, chess games are substrate neutral and can be realized in a virtually limitless variety of physical media. Furthermore, in ascertaining whether a given sequence is a legitimate game, all we need to know is whether or not each move is in accordance with the abstract structural rules of chess. The question of *how* these moves were physically accomplished is entirely irrelevant. Was the white bishop picked up and moved with the right hand or the left? Held between thumb and forefinger or thumb and index finger. Or perhaps moved by the power of psycho kinesis? Obviously the answer makes no difference.

2.6 Counterfactual Constraints

The *counterfactual* requirement is aimed at another apparently 'slack' feature incorporated by Putnam and the SMA, *viz*. the mapping from formalism to physical system is defined for only a single run, and says nothing about what *would* have happened *if* a different input had been given. And it is objected that this is too weak to satisfy the more rigorous operational notion of being a 'genuine' realization. However, in response to this quite natural proposal, it is worth noting that for a

physical system to realize a rich computational formalism with proper input and output capacities, such as an abstract TM, this will always be a matter of *mere approximation*. For example, any given physical device will have a finite upper bound on the size of input strings it is able to process, its storage capacities will likewise be severely limited, and so will its actual running time. In principle there are computations that formal TMs can perform which, even given the fastest and most powerful physical devices we could imagine, would take longer than the lifespan of our galaxy to execute. Hence even the fastest and most powerful physical devices we could envision will still fail to support *all* the salient counterfactuals.

So it will never be possible to construct a complete physical realization of an abstract TM – the extent to which a concrete device can execute the full counterfactual range of state transitions of which the abstract machine is capable will always be a matter of *degree*. For example, consider the exceedingly simple machine ***M*** given above. It's a straightforward matter to exhibit *particular* computations on small inputs. But there is no finite upper bound on the size of input strings that this abstract machine can handle. The set of four quadruples yields a mathematically well defined and effective procedure for adding one to a monadic input string which contains in excess of, say, $10^{10000000000000000000000000000}$ 1's. It's not physically possible for *any* artefact that we could build to carry out computations for such astronomically large inputs. Hence *no* physical implementation of this simple three state TM can deal with the full range of *possible* inputs.

So, in general, the class of counterfactual cases on alternative inputs with which a physical realization can cope is by necessity limited – not all counterfactual cases will be supported by *any* physical device implementing any TM. And this renders the appeal to counterfactuals inescapably *ad hoc.* The restrictive strategy demands that the mapping be able to support counterfactual sequences of transitions on inputs not actually given – but precisely *how many* inputs not actually given? One, two, twenty trillion? For any implementation, there will be a finite upper bound on the size of input string it can process, and beyond that size there will be *infinitely many* potential inputs for which it will not be able to perform the salient computation.

This indicates that there is no clear or principled cut off point demarking 'genuine' implementations from 'false' ones in terms of counterfactuals. As another, more common place, illustration of the *ad hoc* nature of the appeal to counterfactuals, consider a standard pocket calculator that can intake numbers up to, say, 6 digits in decimal notion. Is this a 'false' realization of the corresponding algorithm for addition, since it can't calculate $10^6 + 10^6$? It's an *approximate* instantiation which is nonetheless exceedingly useful for everyday sums. It will always be a matter of degree how many counterfactuals can be supported, where a single run on one input is the minimal case. Where in principle can the line be drawn after that? It's a matter of our purposes and goals as interpreters and epistemic agents, and is not an objective question about the 'true' nature of the physical device as an implementation. In some cases we might only be interested in the answer for a single input, a single run.

In addition, Bishop (2009) has importantly extended the SMA strategy to show that any predetermined finite set of counterfactuals *can* be accommodated on this

approach. From this I would conclude that the underlying and more general constraint of concern to those who would delimit the range of physical implementation is neither causal nor counterfactual. Instead, the point to emphasize is that in ↑MR* exercises of this sort, the mapping is entirely *ex post facto*. The abstract procedural 'trajectory' is already known and used as the basis for interpreting various state transitions in the open system and hence characterizing it as an implementation. Hence using this *ex post facto* tactic, even finite sets of counterfactuals can be included. And as emphasized above, our actual computational artefacts are themselves only capable of handling finite sets of counterfactuals.

For a physical device to successfully '*perform* a computation' is distinct from '*fully implementing* a computational formalism'. Performing a computation is an occurrent series of events, an actual sequence of physical state transitions yielding an output value in accord with the normative requirements of the mapping. And this can be satisfied in the case of computing the value of a single output on a given input. In contrast, fully implementing a computational formalism is a much more stringent and hypothetical notion, requiring appeal to counterfactuals, and as above, this will only ever obtain as a matter of degree. In light of this distinction, it is clearly possible for a physical device to successfully perform a computation *without* instantiating a complete computational formalism, which distinction in turn fatally undermines the theoretical force of counterfactuals in attempting to determine whether a physical process has 'really' performed a computation.

2.7 Computational Ascriptions Are Normative

As mentioned above, part of the reason why underlying causal considerations are the wrong level of analysis is that there is no sense in which error or malfunction can occur at this basic physical plane. Physical systems, as such, are governed by *natural laws*, while formal systems are intrinsically *rule governed*. In the case of our computational artefacts, a system governed by natural laws must be deliberately engineered so that it can be interpreted as evolving in accordance with a chosen rule governed formal system. 'Obedience' to natural law is an essentially *descriptive* matter and there is no sense in which mistakes or error can be involved – such laws cannot be broken, and the time evolution of material systems is wholly determined (in the classical case at least) by the regularities in question. On the other hand, 'obedience' to formal rules is an essentially *normativ*e matter, and there is a vital sense in which error and malfunction can occur.

This normativity has nothing to do with ethical or religious considerations, but simply with *conventionally imposed* norms. Suppose we are playing a game of chess. It's my move and it's clear that I'm about to be checkmated by your queen. So I pick up your queen and throw it out the window. You object with the exclamation 'You can't do that!' And I reply, 'What do you mean – I just did'. In this case the physical processes in question are in perfect accord with natural law, but have discontinued implementing the norms of chess. Similarly, if my desk

top machine is dosed with petrol and set on fire while still in operation, the time evolution of the hardware will remain in perfect descriptive accord with natural law. However, it will very soon fail to comply with the normative requirements of implementing Microsoft Word, and serious computational malfunctions will ensue. Being an implementation of Microsoft Word is a normative and *provisional* interpretation of the hardware system, which can be withdrawn when something goes 'wrong' or when the system is disrupted by non-design intended forces – being an implementation of Microsoft Word is not intrinsic to the physical structure itself. It is only at a *non-intrinsic* prescriptive level of description that 'breakdowns' can occur, and we characterize these phenomena as malfunctions only because our extrinsic ascription has been violated (as in Kripke 1982).

Accordingly, I would argue that the status of computation is very different than the status of abstract mathematical theories in physics. In physics we are attempting to give a fundamental characterization of 'reality', and in principle at least all existent phenomena supervene upon this fundamental level. There is no substrate neutrality in this case, and instead we are attempting to arrive at a theoretical description of the fixed and given natural order. So the mapping from abstract formalism to physical values is not purely conventional as with SMA – e.g. the variables are mapped to basic physical magnitudes and not just anything we please. And in the mathematical descriptions of basic physical theory there is *no normativity involved*. If the predictions of a particular theory, say Newtonian mechanics, turn out to be incorrect in certain cases, we do not say that physical reality has therefore 'malfunctioned'. Instead we say that Newtonian mechanics is at fault and our mathematical description *itself* is incorrect.

Imagine that we take a device intended to compute some given arithmetical function. There is always a non-zero probability of error for any algorithm implemented in the physical world – files become 'corrupted', overheating induces processing 'faults', 'errors' are propagated. Since error is always possible it follows that there is no independent fact of the matter regarding which function or algorithm is 'really' being computed. Suppose we say that the device is computing addition. We confirm this by testing its behaviour on 50 thousand inputs and it gives the correct outputs. But unknown to us the device possesses a mechanical fault, and when we keep going it gives some 'wrong' answers for larger inputs. So which function is it *really* computing – addition with errors, or the actual function in extension that corresponds to its physical behaviour? I would say there is no objective fact to the matter. In the arithmetical case there's an extra level of attributed *abstract* computational 'behaviour' that is always *underdetermined* by its actual performance, and which does *not* supervene upon underlying physical microstructure.

According to Piccinini (2015b), one of the prime advantages of the mechanistic approach is that it can account for cases of miscomputation. In this regard it diverges from a purely causal story by invoking normative/functional considerations. However, I would respond that these normative standards are not objective features of physical systems *per se*, but rather are purely conventional human interpretations, on the same par with computational ascriptions themselves. In the case of artifacts, the

mechanistic account must invoke the intentions of the human designers in order to characterize error and malfunction. But this does not successfully address Kripke's philosophical critique, since the purpose and normativity are still entirely in the eye of the human beholder.

In the case of biological systems, including brains, the mechanistic account shifts the burden of the intentional homunculus onto the 'purposiveness' of biological 'design'. According to this type of neo-Darwinian strategy, something has a particular biological function if this function was selected in the course of the organism's evolutionary history. In the present discussion there is not sufficient space to offer a sustained critique of this move. However, in brief I would argue that the attribution of purpose is again just a subjective projection on the part of human theorists, and constitutes a potentially misleading gloss on evolutionary processes. The term 'natural selection' can suggest that some sort of choice mechanism is involved, which can in turn suggest a form of proto-intentionality on the part of biology – as if 'Mother Nature' literally chooses the most fit to survive. But of course this is only a metaphorical take on the fact that possessing some aimlessly mutated trait which just happens to constitute an advantage over ones competitors will mechanically *cause* the possessor to propagate more numerously. The actual mechanisms are all straightforwardly causal, and there is no real need to invoke anthropomorphic heuristics appealing to purpose or design. The operational effect of possessing a randomly generated favorable trait will appear *as if* the trait were 'selected', but of course there is no 'invisible hand' at work. It may be an arch conservative stand in contemporary intellectual culture, but I would still concur with Hume that it's a basic conceptual fallacy to try and derive an 'ought' from an 'is'.

2.8 Computational Ascriptions Are Interest Relative

I would now like to propose a different perspective on the issue. Rather than distinguishing 'true' from 'false' cases of implementation, what the various proposed constraints do instead is to go some distance in distinguishing interesting and *pragmatically useful* implementations from the many uninteresting, trivial and useless cases that abound in the space of theoretical possibility. It's certainly true that there is no pragmatic value in most interpretive exercises compatible with ↑MR and ↑MR*. Ascribing computational activity to physical systems is *useful* to us only insofar as it supplies *informative outputs*.

So, interesting and useful mappings are such that we can directly read-off something that *follows from* the implemented formalism, but which we didn't already know in advance and explicitly incorporate into the mapping from the start. That's the incredible value of our computational artefacts, and it's one of the only *practical* motivations for playing the interpretation game in the first place. Hence a crucial difference between our computational artefacts and the attributions of formal structure to naturally occurring open systems, as employed by ↑MR* exercises, is

that the mapping in the latter case is entirely *ex post facto* and thus supplies us with no epistemic gains. The abstract procedural 'trajectory' is already known and used as the basis for interpreting various state transitions in the open system and hence characterizing it as an implementation. In sharp contrast, we can use the intended interpretation of our artefacts both to *predict* their future behaviour, as well as *discover* previously unknown output values automatically.

And this is obviously why an engineered correlation obtains between fine-grained causal structure and abstract formal structure in the case of our artefacts – we want them to be informative and reliable! We also want them to be highly versatile, and this is where counterfactual considerations can come to the fore in practice: over time we do runs on a huge number of different inputs, and in principle the future outputs follow as direct consequences of the intended interpretation. And this is where semantic considerations can enter the picture – the purely syntactic formalisms are designed to *preserve truth* in our intended interpretation, so that from the automated syntactic transformations we can apply our interest-relative semantics and hence discover new truths about our chosen semantic domain. In general, a particular physical device is *useful to us* as a computer only when its salient states are distinguishable by us with our measuring devices, and when we can put the system into a selected initial state to compute the output of our chosen algorithm on a wide range of input values. And these features will be relative to our current technological capabilities.

These *pragmatic* considerations supply clear and well motivated criteria for differentiating useful from useless cases of physical implementation. And I would advocate this type of pragmatic taxonomy in lieu of attempts to give overarching theoretical constraints purporting to distinguish literally 'true' from 'false' cases. The pragmatic factors do not supply global and uniform necessary conditions (and the ever present non-zero probability of error indicates than none is *sufficient*, either). Different desiderata will have shifting roles and prominence in different contexts of application, and will be satisfied to varying *degrees* dependent on the goals and purposes in question, as well as the state of our technological progress. Computation is a highly versatile tool, and there is no single and objective class of phenomena that can be isolated as comprising the 'real' instances of physical implementation. Instead, SMA specifies the maximal and context neutral space of possibilities, and varying pragmatic considerations can then be applied to carve out different subsets within this space which prove useful or interesting according to our divergent human purposes. In short, physical computation is not a natural kind – it is founded upon human convention, interpretation and choice.

2.9 Some Standard Objections

I will end the paper by briefly addressing some objections that often arise in response to this position.

2.9.1 The Spectre of Pancomputationalism

In his excellent and illuminating Stanford Encyclopedia article, Piccinini (2015a) observes that one of the motivations for rejecting SMA is that it induces 'unlimited pancomputationalism', which is presumably something we should wish to avoid. But it's difficult to see why this type of pancomputationalism should constitute a theoretical menace, since it goes hand in hand with anti-realism about physical computation, and simply implies that any number of abstract mappings exist in *a purely mathematical sense*. Analogously, there are any number of abstract mappings that exist from the set of positive integers to collections of physical objects and particles. For example, the set of O_2 molecules in some arbitrarily delimited region of the atmosphere is enumerated via some function on the positive integers. And this region can be defined as a proper subset of some other region and the same molecules are enumerated by any number of different functions. Hence the same molecules can be members of arbitrarily many different sets and images under many different mappings. Is this a threat? For the most part we don't care about all these possible sets and enumerations. But in some cases we do, as in the set of human beings living in some country, when it comes time to do a census.

2.9.2 The Threat to CTM

I endorse a purely formal and non-intrinsic account of computation, and consequently argue that the mathematical theory of computation alone is not sufficient to provide a full explanatory *theory of* particular subject disciplines, such as a *computational theory of the mind*. This is a specialized scientific application that requires many additional resources appropriate to the phenomena and subject area under investigation. Computation is an extremely powerful and versatile formal tool, that can be applied to a virtually limitless range of phenomena. However computation *per se* has no *mystical powers*, and merely implementing the 'right' sort of computational formalism cannot magically transform some given arrangement of mass/energy into a mind. On my account, much more is required than merely implementing a formal procedure. In particular, the system must be able to *do* a host of complex and sophisticated things within a multifaceted environment. See Schweizer (2016) for further discussion.

2.9.3 Not All Levels of Description Are 'Intrinsic' from the Perspective of Physics

There are many levels of description that are not 'intrinsic' from the perspective of fundamental physics, but are nonetheless perfectly legitimate and scientifically

respectable. For example, various arrangements of mass/energy configured in such a way as to perform some clear biological function, such as 'being a kidney'. In response, I would argue that the attribution of computational structure is crucially *disanalogous* to cases such as this, which still trade on characteristics which are themselves essentially physical in nature. In order to be a kidney, a particular assemblage of material stuff must *do things* with other instances of material stuff that are characterized in terms of, e.g. the chemical composition of blood, waste products, filtering, etc. There is an objective, observer independent fact of the matter regarding whether or not a given configuration of matter performs the chemically specified functions required of kidneys, because biological functions are defined in terms of cause and effect relations in the physical world, and in stark contrast, computational realizations are *not*.

There is a pronounced difference here between *actual* versus *abstract* characteristics which makes attributions of computational structure observer dependent in a manner not shared by biological functions. The inputs to a computational system are essentially 'symbolic' rather than physical, where the material implementations of the symbolic or formal inputs must be *interpreted* as such by an outside agent, and where this symbolic interpretation is entirely *conventional* in nature. This marks a prominent discontinuity in levels of description.

2.9.4 There Are Objective Constraints If Given an Appropriate Physical Description

Not just anything goes as SMA seems to suggest – there *are* objective constraints at appropriately specified levels of physical description, e.g. circuit theory (see Scheutz 1999). And I would agree that, relative to particular design parameters imposed by human engineers, in conjunction with known principles of materials science, there can be very tightly constrained abstract solutions. SMA does not imply that such mappings are 'arbitrary', and surely the impressive success and reliability of our artifacts is not a subjective phenomenon. As with Dennett's Intentional Stance, predictive success is an objective criterion. However, to the extent that success *is* achieved, it ultimately rests upon skilled manipulation of the physical substrate. And the ever present possibility of error and malfunction indicates that an abstract computational description of this (continuous) substrate is still a normative idealization and not an 'intrinsic' characterization. There is nothing physically or metaphysically privileged about circuit theory as a level of description, and it does not preclude alternative characterizations and different computational mappings ascribed to the very same physical system. Hence such 'favored' mappings have no impact on the basic SMA perspective.

2.9.5 SMA Cannot Differentiate a Stone from a Sophisticated Computational Artifact

And surely there *is* a difference, objectors will contend, and hence SMA does not provide a satisfactory account of computation in physical systems. To this complaint I would reply that the crucial difference is in our ability to manipulate the artifact in order to acquire new information. Artifacts are specifically designed and built to satisfy non *ex post facto* mappings – this is why they're so useful and why we pay good money for them. But this feature does not ground an ontological distinction between 'real' versus 'spurious' implementations. In other cases we appeal to *ex post facto* methods, as in error checking the very same artifacts. And in the case of 'natural computation', if we have a theory concerning what computation a given biological system is performing, then we can predict future *physical* states of the system, and also test our theory, by carrying out the computation *first* and then looking to see if it maps to the empirical facts.

References

Bishop, J.M. 2009. Why computers can't feel pain. *Minds and Machines* 19: 507–516.

Block, N. 2002. Searle's arguments against cognitive science. In *Views into the Chinese room*, ed. J. Preston and J.M. Bishop. Oxford: Oxford University Press.

Boolos, G., and R.C. Jeffrey. 1989. *Computability and logic*. 3rd ed. Cambridge: Cambridge University Press.

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

Chrisley, R.L. 1994. Why everything doesn't realize every computation. *Minds and Machines* 4: 403–420.

Copeland, J. 1996. What is computation? *Synthese* 108: 335–359.

Dennett, D. 1981. True believers: the intentional strategy and why it works. In A. F. Heath (Ed.) *Scientific Explanation: Papers Based on Herbert Spencer Lectures given in the University of Oxford*, Oxford: University Press.

Fodor, J. 1981. The mind-body problem. *Scientific American* 24: 114.

Kripke, S. 1982. *Wittgenstein on rules and private language*. Cambridge: Harvard University Press.

Maudlin, T. 1989. Computation and consciousness. *Journal of Philosophy* 86 (8): 407–432.

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

Newman, M. 1928. Mr. Russell's "Causal Theory of Perception". *Mind* 37: 137–148.

Piccinini, G. 2015a. Computation in physical systems. In *The Stanford encyclopedia of philosophy,* ed. E.N. Zalta. http://plato.stanford.edu/archives/fall2015/entries/computation-physicalsystems/.

———. 2015b. *Physical computation*. Oxford: Oxford University Press.

Putnam, H. 1988. *Representation and reality*. Cambridge: MIT Press.

Rescorla, M. 2014. A theory of computational implementation. *Synthese* 191: 1277–1307.

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

Schweizer, P. 2012. Physical instantiation and the propositional attitudes. *Cognitive Computation* 4: 226–235.

———. 2016. In what sense does the brain compute? In *Computing and philosophy*, Synthese library 375, ed. V.C. Müller, 63–79. Heidelberg: Springer.

Searle, J. 1990. Is the brain a digital computer? *Proceedings of the American Philosophical Association* 64: 21–37.

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

Sprevak, M. 2010. Computation, individuation, and the received view on representations. *Studies in History and Philosophy of Science* 41: 260–270.

Turing, A. 1936. On computable numbers, with an application to the entscheidungsproblem. *Proceeding of the London Mathematical Society*, (series 2) 42: 230–265.

———. 1950. Computing machinery and intelligence. *Mind* 59: 433–460.