Computational Individuation

Abstract

I argue that accounting for computational individuation is the most important desiderata of a theory of physical computation by showing that indeterminacy objections to structural/mechanistic accounts of computation reduce to the indiscernibility objections made against mathematical structuralists. Roughly, this is because the structural *invertibility* of logic-gates such as AND/OR is caused by the structural *identity* of the binary computational digits 0/1 themselves. I use a proof of this result to show that pure computational structuralism is untenable because structural indeterminacy entails absurd consequences – namely, that there is only one binary computational digit.

§1. When it comes to providing a satisfactory account of physical computation, three main adequacy conditions emerge from the literature. The first and most classic is the avoidance of triviality, i.e., ensuring one's theory does not entail that every physical system implements every computation.[1] The second adequacy condition is that computational explanation is medium independent, i.e., that an account of physical computation can capture the fact that computational systems are built with distinct materials and specifications.[2] The third most recently discussed condition that is primarily aimed against computational structuralism, is to ensure the determinacy of a truth-functional implementation of a structural dual pair, i.e., that an account can tell us when a physical system is computing AND rather than OR.

I will argue that failing to meet this third adequacy condition is far more problematic than has been recognised since it can lead to a *reductio ad absurdum*. This demonstrates that ensuring the determinacy of a truth-functional implementation is the most important of the three adequacy conditions, since failing to meet the others, though objectionable, does not entail *absurdity*. After all, legitimate views like pancomputionalism reject the first (e.g. Scheutz [2001]) and some structuralists reject the second (e.g. Dewhurst [2018b]).

[1] Note this is the weakest version of the triviality condition; it can be strengthened to ensure an adequate theory should not entail that every physical system implements *some* computation, i.e., the theory should place suitable restraints on the simple mapping account. See Putnam [1988]; Sprevak [2018]; Schweizer [2019].

[2] This can be understood as a species of Putnam's multiple realisability thesis in the philosophy of mind; a single computational state can be realised by many distinct physical systems (provided the systems' physical properties can support the state-transition rules, etc.). See Putnam [1967]; Shapiro [2000].

First I'll show that for 'pure' computational structuralism a reductio can be established by means of a simple mathematical proof (§3-4). I'll then survey the prospects for modern 'hybrid' structuralism and conclude that the threat of the reductio will, at best, force them to foreclose on satisfying the second adequacy condition: medium independence (§5-7).

§2. The third adequacy condition, which I shall call the *determinacy* condition, was first raised by Shagrir [2001, 2012] as an objection to structuralist accounts of physical computation – which broadly hold that physical computation is determined by the causal/functional/mechanistic structure of the physical system.[3] The simplest version of the objection, due to Sprevak, is given by the 'duality' of basic Boolean gates. That is to say, the fact that pairs of two-input single-output gates such as AND/OR are invertible, such that appealing to their structural features alone cannot hope to determine whether a given physical component is an AND-gate or an OR-gate. For an illustration of this, Sprevak [2010, 296] gives a simple gate which is sensitive to voltage ranges 0–5v (0) or >5v (1):

input 1	input 2	output
1	1	1
1	0	0
0	1	0
0	0	0

Table 1: Gate 1

Blatantly, Gate 1 could be used to compute either AND or OR because the assignment of the voltage ranges 0/1 to truth values T/F *must be arbitrary if our only recourse is to the structural features of the physical system.* This indeterminacy is not restricted to AND/OR; it generalises to any structural dual pair of two-input single-output Boolean gates across different physical mediums (e.g. electric, hydraulic). For example, NAND/NOR, XOR/XNOR.[4]

Note that on the main alternative view to structuralism, semanticism, this indeterminacy will be resolved by the additional representational content which semanticists characteristically appeal to beyond the merely structural features of the physical system.[5] Thus Gate 1 can be said to implement AND if the voltage range 0 *represents* F. For their part, most structuralists – such as Dewhurst [2018b], Coelho Mollo [2018], Miłkowski and Fresco [2019] – opt to bite the bullet with respect to this indeterminacy and accept the underdetermination of physical computation

[3] See Egan [1992]; Chalmers [1996]; Miłkowski [2013]; Fresco [2015] & Piccinini [2007, 2015].
[4] See Shagrir [2001]; Miłkowski and Fresco [2019, 2].
[5] Classic semantic accounts include: Dennett [1971]; Fodor [1998]; Shagrir [2001, 2012]; Sprevak [2010]; Rescorla [2014].

as a fact of structuralist life. After all, they can still maintain there is *some* mapping between the formal and the physical. Even a surjective non-injective mapping would guarantee that Gate 1 is mapped to a logical function – just not uniquely. Thus, the structuralist need not forgo the possibility of computational analysis when conceding to the semanticist that certain truth-functions are systemically underdetermined by structuralist resources.

In what follows, I will argue this indeterminacy is caused by a more fundamental indeterminacy and is far more problematic than either semanticists or structuralists have realised, since it commits structuralists to fatally absurd, and not merely indeterministic, results. In particular, the equivocation of the two digits of binary computational 0/1.

§3. Before we go further, I must draw a distinction between 'pure' structuralist accounts which appeal only to structural features of the causal system, such as can be found in Chalmers [1996, 2011], Dewhurst [2018a], Schweizer [2019] and 'hybrid' structuralist accounts which rely on an appeal to non-structural features such as mechanisms or telofunctions, such as Piccinini [2015], Coelho Mollo [2018], Miłkowski [2013] & Fresco [2015]. This distinction is important because although the reductio can be raised against both species of structuralist accounts, they require separate treatment for reasons that will become clear. Until §5, my focus will be on pure structuralist accounts for which the reductio's application can be proven formally.

My first conjecture is that if there is indeterminacy with respect to the computational truth-functions, then there is an (intractable) indeterminacy with respect to the computational digits 0/1. By digit, I will mean the most fundamental computational individuals given by a physical system under a formal interpretation. Most views do not take all of the properties of physical states (such as voltage levels) to be computationally relevant. The relevant properties of the physical states are conventionally represented by Boolean values (as in table 1). For pure structuralists, the computationally relevant properties are exhausted by the causal structure of the physical states within the context of the system, i.e., the digit's structural profiles. In this way, I distinguish physical states (e.g. 0V/5V) from computational digits (0/1) from truth-values (T/F). Thus, digits are not abstract states but slight abstractions from the physical states, or again, representations/types of the computationally relevant features of the physical states implementing the computation.

To set out the indeterminacy between the computational digits, we return to Gate 1 (Table 1). Consider the following: if the digits 0/1 in Gate 1 were determinate, then the truth-functions could not be indeterminate, e.g., if we assigned 0 to F, Gate 1 would implement AND. Contrapositively, if there *is* indeterminacy in the truth-functions then there cannot be determinacy in the digits. Therefore, the indeterminacy of AND/OR issues from the underlying indeterminacy of 0/1. It seems no structural features of the voltage ranges 0–5v/>5v *could* determine which range should be assigned T/F. This will be the case for all problematically structurally invertible duals because the key point here is that computational truth-functions are *truth-functional*, i.e., they are exhaustively defined by their truth tables such that their values are a function of their digit input.

Structuralists who accept the indeterminacy of the truth-functions will be unmoved by the indeterminacy of the digits and wonder why they cannot simply adopt a many-one relationship between the logical values and physical states. My next conjecture shows why not: if the computational digits are indeterminate, then they are structurally identical. If the structural profiles of the computational digits are *identical*, rather than merely *invertible*, this commits the structuralist to the claim that there is only one computational digit – an absurdity so great it must forfeit the very legitimacy of the account, undermining, as it does, any coherent conception of *binary* computation.

Structural identity is standardly established by demonstrating that some element or function can be permuted while the structure of the domain, or system, is preserved. As such, one only need reflect on the fact that the voltage ranges the computational digits represent can be swapped without change to the computational system to see that the digits are structurally identical and hence identical for a pure structuralist. This point, which threatens to draw from pure structuralism a consequence absurd enough to refute it, admits of a proof given below.

§4. To prove structural identity mathematically we standardly define a structure preserving permutation, i.e., a non-trivial automorphism. For physical computation this amounts to defining a gate in a given computational system which (determinately) computes a truth-function which permutes the digits, i.e., an implementation of a non-trivial automorphism. Given a simple computational system with three logical connectives (NOT, AND, OR) we can define the automorphism with the following gate, where 1/0 represents the structurally relevant feature of some specifiable discrete physical states:

input	output
1	0
0	1

Table 2: Gate 2

Let S be the two-membered set of physical states including discrete voltage ranges (say S = {0–5v, >5v}). The function f implemented by Gate 2 can then be defined:

$f : S \to S$ given by $f(x) = \neg x, x \in S$.

Let us now prove that f is a non-trivial automorphism: Since f is not the identity function $f(x) = x$, f is non-trivial. A function is an *automorphism* iff it is an isomorphism which maps the set S to itself. A function is an *isomorphism* iff it is a bijection and a homomorphism. A function is a *bijection* iff it maps to every element in the set uniquely, i.e. it is surjective and injective. It is straightforward to prove that f is a bijection by the fact that the function simply swaps 1 and 0 by replacing each digit (surjective) with the other (injective). However, f is less obviously homomorphic. A function is a *homomorphism* if it is a structure-preserving mapping, i.e., a function h such that for the sets G, H under operations $(G, \sim)$ and $(H, *)$, x, y ∈ G, $h : G \to H : h(x \sim y) = h(x) * h(y)$. Since we are establishing an *auto*morphism, we need to show for $(S, \sim)$ and $(S, *)$, x, y ∈ S,

$f : S \to S : f(x \sim y) = f(x) * f(y)$

for each of the operations defined on S, i.e. NOT, AND, OR. It is straightforward to show that $f(\neg x) = \neg f(x)$. Substituting $f(x) = \neg x$ gives: $\neg(\neg x) = \neg f(x)$ / $\neg\neg x = \neg(\neg x)$. Next, although $f(x \wedge y) \neq f(x) \wedge f(y)$, we can prove: $f(x \wedge y) = f(x) \vee f(y)$ / $f(x \vee y) = f(x) \wedge f(y)$

by De Morgan's Laws since substituting $f(x) = \neg x$ gives

$\neg(x \wedge y) = \neg x \vee \neg y$

$\neg(x \vee y) = \neg x \wedge \neg y$.

Revealingly, this latter part of the proof – i.e., that the operations OR/AND preserve each other's structure across an automorphic mapping – is a mathematical way of formulating the original indeterminacy objection as raised by Sprevak.

Since f is a bijection and homomorphism, f is an isomorphism. Since f is an isomorphism which maps S to itself and not merely the identity mapping, the function implemented by Gate 2 is a non-trivial automorphism. The significance of this result is that it serves as a formal articulation and proof of the conjecture that the computational digits 0/1 have an identical structural profile. It works by precisfying sameness of structure mathematically using some of the basic tools of group theory. Since we chose the set S arbitrarily, the result applies to all binary sets of physical states implementing computational digits without loss of generality.

It is no coincidence that a growing number of opponents of *mathematical* structuralism have defined automorphisms in precisely this way to establish precisely the same thing about various *mathematical* objects – namely that they are problematically structurally identical. Such proofs issue from disparate fields of mathematics, ranging from complex analysis, group theory, and even Euclidean space.[6] The most problematic cases define automorphisms between two unlabelled nodes in a graph.[7] However, the classic example is an automorphism between $a + bi$ and $a - bi$ on the complex field; i.e., a function $f : (\mathbb{C}) \rightarrow (\mathbb{C})$ given by $f(x) = -x, \forall x \in \mathbb{C}$.

§5. To take stock, the reductio and its proof establish our two conjectures: that if pure structuralists accept the indeterminacy of structural duals, then they are committed to the indeterminacy of computational digits; and if they are committed to the indeterminacy of computational duals, they are provably committed to their identity, which is nonsense. Therefore, pure structuralists cannot continue to bite the bullet when it comes to the determinacy condition.

Not only does this provide a new kind of objection to pure computational structuralism, but it sheds an important light on the determinacy condition itself. In fact, I think it merits a complete reformulation of the condition. For it is now clear that in order to provide a determinate account of computing logic gates, we must be able to individuate the fundamental computational digits, reducing the *determinacy* problem to the problem of providing an adequate account of computational *individuation*, not merely for truth-functions.

As we have seen, extra-structural resources are required to distinguish the digits contra pure structuralism. This brings us to hybrid resources such as Piccinini's proper functions or Coelho Mollo's telofunctions. Hybrid accounts are, for the most part, safe from the formal proof of the reductio because they import formally intractable appeals to such mechanisms and functions. Vitally, however, it is incumbent on hybrid accounts to precisify the concepts they use to characterise computation – if not formally – to the extent that they can provide a criteria of individuation for computational digits. Otherwise, such accounts will be both immune to reductio proof and impotent to satisfy the individuation condition simply because the concepts they import are too vague. Therefore, hybrid accounts are still vulnerable to the reductio if their particular account of individuation is not able to distinguish digits, though this can only be established on a case-by-case basis.

Unfortunately, of all the adequacy conditions, comparatively little work has been done on

[6] See: Burgess [1999]; Shapiro [2008, 2012]; MacBride [2006]; and Ladyman [2005].
[7] Leitgeb and Ladyman [2008, 390–93]

structuralist accounts of individuation. The arguments I have provided for the importance of this condition and the disastrous consequences of neglecting it will hopefully put this to right, but the burden of proof here lies squarely with contemporary structuralists. Two laudable exceptions include Dewhurst [2018b] and Coelho Mollo [2018], so I want to finish by surveying the prospects of their respective attempts to provide an account of individuation by non-structural non-semantic means.

§6. Dewhurst's key insight is his distinction between two criteria of individuation operating on computational systems: *algorithmic equivalence* and *computational equivalence*. The former is grounded in logical equivalence and the latter in physical equivalence such that the same logical function can be computed by distinct computational systems [Dewhurst, 2018b, 110].[8] On Dewhurst's account the indeterminacy will remain because it maybe be indeterminate which logical function a computational system is computing. However, he avoids the reductio as follows: the definition of the automorphism establishes neither the *algorithmic* equivalence of the digits nor their *computational* equivalence. This is because the structurally identical digits 0/1 will be kept algorithmically distinct in virtue of their algorithmic equivalence being grounded in the distinctness of the truth-values T/F. Similarly, the digits will be kept computationally distinct in virtue of their computational equivalence being grounded in two distinct physical states, e.g., 0–5v/>5v. In this way, Dewhurst's fix meets our new condition by tendering out the individuation of the computational digits to the identity criteria of the physical states.

Unfortunately, Dewhurst's proposal globally undergenerates computational equivalences.[9] The very same mechanism which protects his account against my objection – i.e., grounding computational identities in physical identities – also entails that, in practise, no two systems are computationally identical due to their inevitable minuscule physical variations.[10] It is thus unclear that Dewhurst is providing an account of *computational* equivalence, given that we cannot capture cases where we want to say that the same computation is being carried out by different physical systems. Therefore, Dewhurst's account satisfies individuation at the cost of another important adequacy condition: the medium independence of computational explanation.

[8]Dewhurst means to supplement Piccinini's mechanistic account of computation. Piccinini's own solution, that a device may implement a multiplicity of computations but that his systemic functions along with the wider system will determine which function is relevant, will not avoid the reductio. Piccinini accepts the indeterminacy which leaves him vulnerable to the indeterminacy and identification of the digits even before we consider he cannot account for fully dual systems.

[9]As pointed out by Miłkowski and Fresco [2019], but first pointed out by Dewhurst himself [2018b, 110].

[10]Within a system there may be no equivalence between processors performing the same algorithmic operation.

§7. Coelho Mollo [2018] and Miłkowski and Fresco [2019] have recently argued Dewhurst's fix can be itself fixed to recapture medium independence on a mechanistic account. This is important because so far it looks like structuralists will be systemically unable to fulfil all the desiderata of an adequate account of computation. Coelho Mollo follows Dewhurst in drawing a distinction between *algorithmic equivalence* and *computational equivalence*.[11] However, he grounds the latter not in physical structure but in "computationally-relevant" functional structure of a physical system. The functional structure, he says, is determined by a "teleological function", e.g., the capacity to "perform computations" [Coelho Mollo, 2018, 3495]. This means physically distinct systems can exhibit the same functional structures if they share a target capacity. Hence, the medium independence of computation is restored on this account.

To take his example, two devices, D1 and D2, with slightly different voltage ranges (0-4/5-10V for D1 and 0-5/6-10V for D2) will have the same input-output tables "when put in terms of equivalence classes" [2018, 3494], as below:

input 1	input 2	output
EC1	EC1	EC1
EC1	EC2	EC1
EC2	EC1	EC1
EC2	EC2	EC2

Table 3: Input–output table of D1 and D2's functional EC's

Although D1 and D2 are physically distinct, the computationally-relevant functional profiles of their input-output equivalence classes (EC's) are identical and hence they count as computationally equivalent [2018, 3496]. Coelho Mollo thus provides a precisification of 'function' tractable enough to provide a criterion of individuation of logic-gates. However, when used for individuation of digits – which are, in this case, EC's – the criterion fails to avoid the reductio.

According to Coelho Mollo the identity of the EC's is defined by the uniform sensitivity of the processing device with respect to its inputs and outputs [2018, 3494]. D1/D2 are sensitive to physically distinct voltage ranges but the functional profiles of those voltage ranges are the same, as in Table 3. This is what justifies him in equivocating EC's defined relative to physically distinct devices, like EC1 of D1 (0-4V) and EC1 of D2 (0-5V). This means of satisfying the individuation condition will be vulnerable to the reductio if the functional profiles of distinct EC's can be shown to be identical. We cannot mathematically prove this because an automorphism would establish the digit's structural *algorithmic* identity, not their functional-structural

[11] My argument will apply equally to Miłkowski and Fresco's account.

computational identity. Instead, we must show that the criterion of individuation used in Table 3 overgenerates to falsely equivocate EC1 and EC2.

Consider a fully dual system containing D1. Let EC1 $= \{0-4V\}$, EC2 $= \{5-10V\}$ and let R be the equivalence relation by which Coelho Mollo equates EC1 of D1 and EC1 of D2 (Table 3). R holds between EC1/EC2 iff EC1/EC2 have identical functional profiles. To show EC1/EC2 have identical functional profiles, observe that EC1/EC2 can be permuted without change to their functional profiles in D1 (table 4).

input 1	input 2	output
EC2	EC2	EC2
EC2	EC1	EC2
EC1	EC2	EC2
EC1	EC1	EC1

Table 4: Input–output table of D1's functional EC's after permutation

The functional profiles of EC1/EC2 are identical. Therefore R holds between EC1/EC2, which is false.[12] Permuting EC1/EC2 can have *no effect on the uniform sensitivity of the processing device*. The EC's are functionally as well as structurally symmetric because any functional differences between EC1/EC2 "play no role in their general computational capacities" [2018, 3496]. Hence, the functional profiles of EC1/EC2 are identical and since computational individuation is wholly determined by the computationally-relevant functional profiles of input-output equivalence classes, EC1 = EC2.

To the pure structuralist, we said that since the digits have, by mathematical proof, identical structural profiles and since appeal to structure is the only means they have of individuating them, they are forced to identify the binary digits, which is absurd. To Coelho Mollo, we say that since the EC's have, by a simple permutation argument, identical functional profiles, and since functional appeal is the only means he has of individuating them, he is forced to identify the binary equivalence classes, which is absurd.

To avoid this absurdity, it seems the hybrid structuralist is forced to retreat to Dewhurst's original proposal of grounding computational individuation in the physical states. This would individuate EC1/EC2 since they are implemented by distinct voltage ranges. As we saw, this gives up on the medium independence condition which threatens the account with explanatory inadequacy. However, since explanatory inadequacy is a far better problem than absurdity, Coelho Mollo's improvement on Dewhurst fares far worst than Dewhurst's original proposal.

[12] Note that we are not merely permuting the names of the EC's (which are of course arbitrary) but the *equivalence classes themselves*, i.e., the digits implemented by 0-4/5-10V.

This is no accident. As Coelho Mollo himself points out, there is an inherent tension between the individuation condition and the medium independence condition. The digits – which are in the binary case just symmetric images – must be fine-grained enough not to be identified but course-grained enough to encompass computation across different mediums. This tension will temper all hybrid accounts and should make us pessimistic at best that structuralists can meet all the criteria of an adequate account of physical computation.

§8. We have established several interesting results: that if computational functions are indeterminate, computational digits are indeterminate; that the indeterminacy of the computational digits implies their structural identity; that the latter result admits of a mathematical proof; that this indiscernibility problem threatens structuralism with reduction to absurdity; that computational and mathematical structuralists face the same objection; that pure computational structuralism is untenable; that for the best available hybrid account, the permutation of the EC's preserves their functional profiles hence showing the EQ's to be identical; and that the burden of proof lies with other hybrid structuralists to urgently precisify their appeals to mechanistic/teleofunctional resources far enough to assess whether their means of individuating computational digits are also vulnerable to such a reductio. Most of all, I hope to have demonstrated that providing an account of computational individuation presents us with an adequacy condition more deserving of attention than that of triviality.

References

Burgess, J. (1999). Review of Shapiro (1997). *Notre Dame Journal of Formal Logic 40*, 283–91.

Chalmers, D. J. (1996). Does a Rock Implement Every Finite-State Automaton. *Synthese* (108), 309–333.

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

Coelho Mollo, D. (2018). Functional Individuation, Mechanistic Implementation: The Proper Way of Seeing the Mechanistic View of Concrete Computation. *Synthese* (195), 3477–3497.

Coelho Mollo, D. (2019). Are There Teleological Functions to Compute? *Philosophy of Science 86*(3), 431–452.

Dennett, D. C. (1971). Intentional Systems. *The Journal of Philosophy* (68), 87–106.

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

Dewhurst, J. (2018b). Individuation Without Representation. *The British Journal for the Philosophy of Science 69*(1), 103–16.

Egan, F. (1992). Individualism, Computation, and Perceptual Content. *Mind* (101), 443–459.

Fodor, J. A. (1998). *Concepts*. Oxford: Blackwell.

Fresco, N. (2015). Mechanistic Computational Individuation. *Erkenntnis* (80), 1031–53.

Ladyman, J. (2005). Mathematical Structuralism and the Identity of Indiscernibles. *Analysis 65*(3), 218–21.

Leitgeb, H. and J. Ladyman (2008). Criteria of Identity and Structuralist Ontology. *Philosophia Mathematica 16*(3), 388–396.

MacBride, F. (2006). What Constitutes the Numerical Diversity of Mathematical Objects? *Analysis 66*(1), 63–69.

Miłkowski, M. (2013). *Explaining the Computational Mind*. Cambridge, MA: MIT Press.

Miłkowski, M. and N. Fresco (2019). Mechanistic Computational Individuation without Biting the Bullet. *The British Journal for the Philosophy of Science* (0), 1–8.

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

Piccinini, G. (2008). Computation without Representation. *Philosophical Studies 137*(74), 205–241.

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

Putnam, H. (1967). Psychological Predicates. In W. H. Capitan and D. D. Merrill (Eds.), *Art, Mind, and Religion*, pp. 37–48. Pittsburgh: University of Pittsburgh Press.

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

Rescorla, M. (2014). The Causal Relevance of Content to Computation. *Philosophy and Phenomenological Research* (88), 173–208.

Scheutz, M. (2001). Causal versus Computational Complexity. *Minds and Machines 11*(4), 534–566.

Schweizer, P. (2019). Triviality Arguments Reconsidered. *Minds and Machines 29*, 287–308.

Shagrir, O. (2001). Content, Computation and Externalism. *Mind 110*(438), 369–400.

Shagrir, O. (2012). Computation, Implementation, Cognition. *Minds and Machines* (22), 137–48.

Shapiro, L. A. (2000). Multiple realizations. *The Journal of Philosophy 12*(97), 635–654.

Shapiro, S. (2008). Identity, Indiscernibility, and *ante rem* Structuralism: The Tale of i and $-i$. *Philosophia Mathematica 16*(3), 285–309.

Shapiro, S. (2012). An 'i' for an i: Singular Terms, Uniqueness, and Reference. *Review of Symbolic Logic 5*(3), 380–415.

Sprevak, M. (2010). Computation, Individuation, and the Received View on Representation. *Studies in History and Philosophy of Science Part A 41*(3), 260–70.

Sprevak, M. (2018). Triviality Arguments About Computational Implementation. In M. Sprevak and M. Colombo (Eds.), *Routledge Handbook of the Computational Mind*, pp. 175–191. Routledge: London.