Marcin Miłkowski

Institute of Philosophy and Sociology

Polish Academy of Sciences, Warsaw

Center for Philosophical Research, Warsaw

Is computationalism trivial?

In: Gordana Dodig Crnkovic and Susan Stuart (eds.), Computation, Information, Cognition – The Nexus and the Liminal, Cambridge Scholars Publishing, 2007, pp. 236-246.

Abstract

In this paper, I want to deal with the triviality threat to computationalism. On one hand, the controversial and vague claim that cognition involves computation is still denied. On the other, contemporary physicists and philosophers alike claim that all physical processes are indeed computational or algorithmic. This claim would justify the computationalism claim by making it utterly trivial. I will show that even if these two claims were true, computationalism would not have to be trivial.

First, I analyze the vague definition of computationalism. By showing how it depends on what we mean by "a computational process", I distinguish two main flavors of computationalism claim:

1. That cognitive processes could be described algorithmically (in G. Chaitin's sense of "algorithmic")

2. That cognitive processes are algorithmic or computational (they implement recursive functions).

This second claim could be analyzed further as a claim:

1. That cognitive processes could be described as computational

2. That cognitive processes are really implemented computationally

3. That cognitive processes are generated by computational processes.

I distinguish then three varieties of computationalism. The first is that cognitive processes can be simulated computationally; the second is that they can be realized computationally; the third is that cognitive processes are generated by overall computational processes. This last sense is on the verge of being trivial if we accept that all physical processes are computational.

I show that the non-trivial computationalism involves a multi-level model of cognition where certain level of organization of processes is emergent on the base level. This base level could be even conceived of as algorithmic but the emergent computational level would implement other algorithms than the base level. I try to sketch a multi-level model of cognition which involves computation without being at the same time trivial.

IS COMPUTATIONALISM TRIVIAL?

In this paper, I want to deal with the triviality threat to computationalism in cognitive science. On the one hand, the controversial and vague claim that cognition essentially involves computation is still denied as false or explanatory vacuous (see for example Searle 1992). On the other hand, many contemporary physicists and philosophers alike accept *universal computationalism* (known also as *pancomputationalism*) – a claim that all physical processes are indeed computational or algorithmic (Wolfram 2002, Fredkin 2005, Lloyd 2000, Ng 2001, Chaitin 2005, and in the context of evolutionary theory Dennett 1995). Universal computationalism would easily justify mental computationalism by making it utterly trivial. I will show that even if all processes are computational and cognition is essentially a computational process, a version of computationalism in cognitive science and philosophy of mind would not have to be trivial.

I will not argue for nor against universal computationalism because such arguments would require a separate analysis of the concepts in question (for a critical discussion of pancomputationalism see for example Piccinini 2007). For the sake of argument, I will assume that some variety of it could be true, and see if it makes mental computationalism completely trivial. The result of my analysis is a taxonomy of possible computationalisms, some of which are weak and trivial, and some of them more robust and non-trivial.

1. What is mental computationalism?

The current usage of “computationalism” is broad and vague. However, all versions of mental computationalism[1] are committed to a claim:

MC) Cognitive processes involve computational processes.

MC could be accepted probably even by Searle 1992 if “involve” is taken here to mean a very weak association relation. There are varieties of mental

1 For the sake of brevity, I restrict myself here to cognitive computationalisms which do not have to imply a claim that all mental processes (for example, emotional or experiential) are also computational. Such hybrid views seem however to be endorsed so they should be accounted for in a full taxonomy of computationalisms.

computationalism which would be acceptable even for vehement critics of the stronger versions of it. I will discuss below three ways to make “involve” more precise.[2]

The second vague term here is “a computational process” or “an algorithmic process”. It could be understood in at least two ways, so I distinguish two main flavors of the computationalism claim:

1. That cognitive processes could be described algorithmically (in G. Chaitin's sense of “algorithmic”, see Chaitin 1975), i.e. they expose non-stochastic regularity which could be accounted for in some compression algorithm.

2. That cognitive processes are algorithmic or computational, i.e. they implement recursive functions[3] or realize computations.

The first flavor of computationalism is a very weak claim that cognitive processes could be described in a scientific theory offering laws (they could be much less strict than physical-mental laws denied by D. Davidson in his anomalous monism, see Davidson 1970). I will focus on a second version which seems more controversial. This one could be broken down further according to the meanings “involve” could have in the original (MC) claim:

1. Cognitive processes could be described as computational.

2. Cognitive processes are really implemented computationally.

2 A similar formulation is to be found in Chrisley 2000, but Chrisley uses “is” instead of “involve”.

3 I am using the notion of recursive functions to define the class of computable functions. This is of course an application of Church/Turing thesis. However, my arguments would also hold if I had used a more neutral formulation with a phrase “... described in terms of the ideal formal computation theory”. Such a move is recommended by Chrisley 2000 as a general defense of computationalism claim. For my purposes, it is not required because I take for granted that universal computationalism has already been defended, and that mental computationalism is understood on today's theoretical grounds because this is exactly what generates the triviality threat. But the universal computationalism in my analysis is exactly transparent in Chrisley's sense: I point only to a possible ideal version of it, and ignore current details.

3. Cognitive processes are generated by computational processes.

These three versions are derived from three possible senses of what an algorithmic process involves:

1) It is a process described in terms of recursive functions (descriptive-algorithmic).

2) It is a process implemented by recursive functions (realization-algorithmic).

3) It is process caused by a process (2) in some physical device (derivative-algorithmic).

The first claims that cognitive processes can be simulated computationally. This is a variety that involves the popular "computer-metaphor" talk. John Searle would not mind endorsing it, as it is quite weak and does not imply that simulated processes are also intentional or conscious. Note that every finite sequence of discrete values is descriptive-algorithmic (Cutland 1980, 122), so if the world is describable by such a finite sequence, it is descriptive-algorithmic, but it is only a trivial definitional implication. This kind of universal computationalism is very weak.

The second claim is that cognitive processes are realized computationally. This is a classical sense of computationalism in cognitive sciences. I will get into necessary details of realization later because on weak renderings of realization, critics of computationalism are tempted to bold suggestions that anything at all computes every possible computable function under some description. I argue that realization of algorithms should be defined not only in terms of discrete states of physical causal processes but of a whole interconnected architecture, and that it is not true that any interpretation goes.

The third variety is a claim that cognitive processes are generated by universal computational processes. This last sense is on the verge of being trivial if all physical processes are computational. But as I argue later, this is not the case for certain models of cognition which stay robust even in a purely digital world.

2. What is realization of algorithms?

A classical account of realization implies that its sufficient and necessary condition is that there is a correspondence relation between states of the program and states of the physical system (see for example Chalmers 1996, 318). The weaker account has it that there should be correspondence between input and output states of the program and of the physical system but such a notion of realization seems to conflate mere algorithmic description with causal implementation.[4] The more robust accounts demand that there should be some structure in between: internal states mediating input/output states. For example, Jerry Fodor requires additionally semantic proof-theoretic relations to obtain:

> Every computational system is a complex system which changes physical state in some way determined by physical laws. It is feasible to think of a system as a computer just insofar as it is possible to devise some mapping which pairs physical states of the device with the formulae in the computing language in such a fashion as to preserve desired semantic relations among the formulae. For example, we may assign physical states of the machine to sentences in the language in such a way that if S_1,…, S_n are machine states, and if F_1, …, F_{n-1}, F_n are sentences paired with S_1, …., S_{n-1}, S_n, respectively, then the physical constitution of the machine is such that it will actually run through the sequence of states only if F_1,…, F_{n-1} constitutes a proof of F_n. (Fodor 1975, 73)

This definition of realization is too broad and too narrow at the same time. Not all programming languages are supposed to be used in a proof-theoretic fashion and only some logical systems have expressibility equivalent to the Universal Turing Machine. We might also add some additional conditions to the specification of a computing language required in this definition, and specifying such a condition can be quite straightforward because this computing language should have expressibility equal to the Universal Turing Machine. This is also a reason why it is more intuitive to directly define required relations between the states of the physical system using Turing machines or other equivalent computation models. Moreover, it is not at all clear if the stipulation of the semantic relations to hold between states of the machine is supposed to exclude all non-symbolic machines, and which machines are taken to be non-symbolic, because Fodor has argued that

4 The reduction of notion of realization to interpretation in both Searle 1992 and Putnam 1988 seems close to such a conflation.

even subsymbolic connectionist networks are symbolic or representational in his sense (Fodor & Pylyshyn 1988).

The other problem is that the correspondence is rendered as a weak condition. Hilary Putnam has notoriously claimed he “proved” that any object realizes any computation (Putnam 1988). Similar objection is to be found in Searle:

> The same principle that underlies multiple realizability would seem to imply universal realizability. If computation is defined in terms of the assignment of syntax, then everything would be a digital computer, because any object whatever could have syntactical ascriptions made to it. You could describe anything in terms of 0's and 1's…. For any program and any sufficiently complex object, there is some description of the object under which it is implementing the program. Thus for example the wall behind my back is right now implementing the Wordstar program, because there is some pattern of molecule movements that is isomorphic with the formal structure of Wordstar. (Searle 1992, 207-208)

David Chalmers in his discussion of this problem admits that *some* computations will be implemented by every system – for example, single element, single-state combinatorial-state automata – but this does not mean that *every* computation is implemented by anything Chalmers 1996, 319). In some ways, one could see objections raised by Putnam and Searle as based on the principle that any ascription of functional properties, especially of formal mathematical properties, is interpretation-based. Would both authors argue also that no objective measurement of physical quantities is possible because measurement scales are interpretation-based? They seem to treat all numerical ascriptions (not only of 0's and 1's) as a matter of interpretation, not as a matter of fact. But at the same time they don't seem to question the validity of all measurements in natural sciences (for a similar objection, see McDermott 2001, 171).

How could one escape such consequences? They would make also digital physics trivial, not only mental computationalism, and they would eventually undermine all uses of mathematics in natural sciences as interpretation-based (or rather interpretation-biased).

First of all, one must require that the syntactical ascriptions to physical objects be consistent. All physical changes in Searle's wall should be isomorphic to changes in a Wordstar program executed on a PC from the beginning of execution until its end: we wouldn't be likely much impressed by a correspondence to a halted Wordstar program looping infinitely.

Mere descriptions do not implement any non-trivial functions (at most, they could be said to compute constant functions) if they are causally inert. This is a common requirement that a whole causal structure of a physical process is as complex as the formal description of the algorithm (Chalmers 1994, 392). It has been however argued that mere causal complexity is not a sufficient condition (Scheutz 2001): to wit, the relation of states of the program being executed to the physical states cannot be sufficiently described with isomorphism which is only defined for functions. So some other relation (like bisimilarity postulated by Scheutz) is required for a physical object to implement computations. I would therefore reject the notion of state-to-state correspondence as too weak.

One other requirement is that computational relations should obtain in machines which are relatively easy to single out from the environment. While there may be a long causal chain which eventually links the internal states of the machine to some distal events, not all causally connected pairs of states of a physical system and its environment should be viewed as parts of the computing machinery (even if one accepts so-called wide computationalism, Wilson 1994, or active externalism, Clark & Chalmers 1998). Thus the system realizing computations should be relatively isolated from its environment so that its computational states could be easily singled out. This condition is that the boundaries of the computational system should be spelled out not in purely computational terms. On a physical level, it is extremely hard to demarcate boundaries of Searle's computing wall, in contrast to the boundaries of a standard PC sitting under my desk, which is a system which can be delineated from the environment using physical and functional descriptions (flow of electricity in the cables etc.).[5] The same objection applies to Putnam's arbitrary disjunctions of physical states which do not form a relatively closed system (see Putnam 1988, 95, for a good discussion see Scheutz 1998).[6]

For this reasons, I define realization as follows:

5 I'm using an informal and intuitive description instead of a formal definition for the sake of brevity. It should be noted that it is complexity of physical relations between the states of a system (which is larger than the complexity of physical relations between the system and its external environment) that underlies the very notion of isolation used here. Nevertheless, a precise definition is not needed for my purposes here.

6 The precise formulation of this requirement is however out of the scope of this article.

> An algorithm *A* is realized in a system *S* iff there is an descriptive algorithmic sequence *A′* (a sequence having a description in terms of recursive functions) encoded in a physical medium that in connection with a physical device *D* causes some derivative algorithmic processes *A''* which in turn generate descriptive algorithmic sequences *A'''*. The encoding must fulfill the requirement of bisimilarity (or similar relation)[7] but is not necessarily discrete, and the system *S* must be relatively isolated from its environment.

Note that the notion of realization as used in digital physics is compatible with the above definition: The cellular automata are relatively isolated systems and they are supposed to implement only a strictly defined set of computations in a given moment. But of course it is a matter of fact if the fundamental physical level is essentially realizing computations in the above sense (this might turn out very hard to show).[8] So while a full definition of realization might seem wordy, it is the price we pay for not accepting Searle's and Putnam's interpretation-only variety of universal computationalism, and at the same time we are not bound to say that all universal computationalism is false by definitional *fiat*.

The stronger universal computationalism makes however the claim “cognition is realized by a computation of an algorithm” trivial. Is there a way to make it more substantial?

3. A multi-level model

A non-trivial computationalism involves a multi-level model of cognition where certain levels of organization of processes are emergent on the base level. This base level could be even conceived of as algorithmic (Wolfram 2002) but the emergent computational level would implement <u>other</u> algorithms than the base level.

There are many notions of emergence in use. I accept here William Wimsatt's concept of emergence as non-aggregativity, where aggregative

7 I don't want to argue for or against any such relation here. I think it is sufficient to point to the kind of relations that would fulfill the task. In this respect, my account is committed to Chrisley's transparent computationalism.

8 It is not so important for my discussion if describing the fundamental physical level in terms of cellular automata is viable.

properties are defined by four conditions:

> (1) a condition on the intersubstitution or rearrangement of parts; (2) a condition on size scaling (primarily, though not exclusively, for quantitative properties) with addition or subtraction of parts; (3) a condition on invariance under the decomposition and reaggregation of parts; and (4) a linearity condition that there be no cooperative or inhibitory interactions among parts in the production or realization of the system property. (Wimsatt 2000)

Wimsatt also defines the critical notion of the organization level: Levels are local maxima of regularity and predictability in the phase space of alternative modes of organization of matter (Wimsatt 1994).[9] Individual levels singled out below are emergent in this very sense.

In a multi-level model of cognition, not all processes, nor all computational processes, could count as cognitive processes. It seems plausible that a multi-level model of a cognitive system comprises at least the following levels:

- Physical and chemical (including quantum level)
- Neurobiological
- Computational
- Representational
- Environmental/Adaptive
- Experiential/Conscious.

A strong multi-level variety of computationalism would be committed to a hypothesis:

> Every cognition is realized by recursive functions which implement algorithms on the internal information processing level of cognitive systems.

So, cognition involves computation in the sense that there is a special computational level realizing special cognitive computations, but even when all physical processes are digital and computational, the emergent higher levels of organization implement other algorithms than the base physical

9 This definition fits my overall strategy to link ways we individuate entities with complexity.

level. The computational level would involve, as in traditional cognitive science, perceptual data processing, memory retrieval etc., but its computations would be implemented by lower level processes or computations. Thus it is not trivial that there is any computational level in cognitive systems.

The crucial point here is that the claims about computational nature stop being trivial when computation is construed of not as a formal relation but rather as the existence of a real-world implementation of a computational architecture in the cognitive system (Sloman 1997). It is interaction of sub-states that makes computation possible. Complex systems such as cognitive systems have architectural complexity that is best described using multiple levels. The exact specification of such an architecture is not a matter of conceptual analysis but rather of a modeling which has to be empirically valid. It may as well turn out that there is a distinct computational level in the architecture, or a whole architecture may turn out computational. This remains a matter of fact, and not of armchair analysis.

4. Summary

Let me review possible versions of computationalism and point at these, which seem free from the triviality threat:

- Weak Regularity Computationalism: Cognitive processes can be described as non-stochastic.
- Weak Simulation Computationalism: Cognitive processes can be simulated as recursive functions.
- Weak Implementation Computationalism: Cognitive processes can be implemented as recursive functions.
- Strong Simulation Computationalism: Cognitive processes are actually simulated (e.g. in animals) as recursive functions.
- Strong Implementation Computationalism: Cognitive processes are actually implemented as recursive functions.
- Weak Multi-Level Computationalism: Cognitive processes could be described as recursive functions on some level of organization of cognitive systems.
- Strong Multi-Level Computationalism: Cognitive processes are implemented by recursive function on some level of organization of cognitive systems.

Only the Strong Multi-Level Computationalism is non-trivial if some robust variety of universal computationalism is true.

The question arises whether this computationalism claim is empirical, metaphysical (conceptual) or simply heuristic for cognitive scientists. The computationalism claim is usually ascribed various statuses: empirical, heuristic, or conceptual. In its purely trivial versions, it is conceptual. In its non-trivial versions, it is also *empirical*, and could play a *heuristic* role. Computational systems are not only systems with some interpretation ascribed intentionally. Their computational structure is as real as any functional structure.

Universal computationalism could make single-level mental computationalism true but trivial. For multi-level computionalism, it is both empirical and conceptual question whether all or some cognitive systems are or could be computational on one of their levels. The hypothesis of the strong multi-level computationalism seems however to be empirical as there are no real conceptual problems with cognitive systems realizing computations, all criticisms notwithstanding.

Works cited

Chaitin, Gregory. 1975. "Randomness and Mathematical Proof". Scientific American, 232, 5, 47-52.

Chaitin, Gregory. 2005. *Meta Math! The Quest for Omega*. Pantheon Books.

Chalmers, David J. 1994. "On Implementing a Computation". Minds and Machines, 4, 391-402.

Chalmers, David J. 1996. *The Conscious Mind. In Search of a Fundamental Theory*. Oxford University Press.

Chrisley, Ron, 2000, *Transparent Computationalism*, in *New Computationalism: Conceptus-Studien 14*, ed. Scheutz, M., Academia Verlag.

Clark, Andy & Chalmers, David J. 1998. "The Extended Mind". Analysis, 58.1, 7-19.

Cutland, Nigel. 1980. *Computability*. Cambridge University Press.

Davidson, Donald, 1970, *Mental Events*, in *Experience and Theory*, ed. Foster, L. and J. W. Swanson, University of Massachusetts Press.

Dennett, Daniel. 1995. *Darwin's Dangerous Idea: Evolution and the Meanings of Life*. Simon& Schuster.

Fodor, Jerry. 1975. *Language of Thought*. Thomas Y. Crowell.

Fodor, Jerry & Pylyshyn, Zenon. 1988. "Connectionism and Cognitive Architecture". Cognition, 28, 3-71.

Fredkin, Ed. 2005. *Introduction to Digital Philosophy* (ms).
Lloyd, Seth. 2000. “Ultimate physical limits to computation”. Nature, 406 (6799), 1047-54.
McDermott, Drew. 2001. *Mind and Mechanism*. MIT Press.
Ng, Y Jack. 2001. “From computation to black holes and space-time foam”. Phys. Rev. Lett., 86, 2946-2949.
Piccinini, Gualtiero. 2007 (in print). “Computational Modeling vs. Computational Explanation...”. The Australasian Journal of Philosophy.
Putnam, Hilary. 1988. *Representation and Reality*. MIT Press.
Scheutz, Matthias. 1998. “Implementation: Computationalism's Weak Spot”. Conceptus JG, 31, 79, 229-239.
Scheutz, Matthias. 2001. “Computational versus Causal Complexity”. Minds and Machines, 11, 543-566.
Searle, John. 1992. *The Rediscovery of Mind*. MIT Press.
Sloman, Aaron, 1997, *Beyond Turing Equivalence*, in *Machines and Thought: The Legacy of Alan Turing I*, ed. Millican, Peter & Clark, Andy, Oxford University Press.
Wilson, Robert A. 1994. “Wide Computationalism”. Mind, 103, no. 411, 351-372.
Wimsatt, William. 1994. “The Ontology of Complex Systems: Levels of Organization, Perspectives...”. Canadian Journal of Philosophy, supp. vol #20, 207-274.
Wimsatt, William. 2000. “Emergence as Non-Aggregativity and the Biases of Reductionism(s)”. Foundations of Science, 5, 3, 269-297.
Wolfram, Steven. 2002. *A New Kind of Science*. Wolfram Media.