Journal of Experimental & Theoretical Artificial Intelligence,
Vol. 15, No. 4, October–December 2003, 375–382

Taylor & Francis
Taylor & Francis Group

Editorial

Subvert the Dominant Paradigm! A review of *Computationalism: New Directions*, edited by Matthias Scheutz (The MIT Press, 2002)

JERRY DEJOHN and ERIC DIETRICH

Department of Philosophy, Binghamton University, Binghamton, New York, USA
e-mails: jdejohn@stny.rr.com, dietrich@binghamton.edu

Abstract. This paper again presses the case for computationalism by considering the latest in ill-conceived attacks on this foundational idea. The authors briefly but clearly define and delimit computationalism and then consider three authors from a new anti-computationalist collection.

1. The rebel yell

Many a rebel youth, denouncing some state of the world, has cried '*Subvert the Dominant Paradigm!*' Unfortunately, few of their number target paradigms that actually need subverting. Some rebels are doing good work, but, in these latter days, many merely wear Rage Against the Machine t-shirts and sport bumper stickers that read 'Kill your TV' and, of course, 'Subvert the Dominant Paradigm'. When encountered at parties, such 'rebel's' never pass up a chance to wax philosophical about the oppressiveness and incompetence of the existing dominant 'thought structures' that form the fabric of our daily existence. If one has the persistence and constitution to ride out one of their diatribes, one usually finds they are attacking the dominant paradigm not because they have correctly identified it as the source of their problems, but because the dominant structure is an easy target for their frustrations. Why subvert the dominant paradigm? Because it is there (to borrow Mallory's phrase), and it presents a much more enticing target than the more elusive prey that is usually the true source of their discontent.

This is exactly the situation with respect to *computationalism*: the hypothesis that cognition is the execution of Turing-computable functions, and the book under review: cognitive scientists have a tremendously useful paradigm that should not be subverted, but it is being attacked by a bunch of Rage-Against-the-Machine-wearing 'rebel's', who, for want of something better to do are out to subvert the dominant paradigm.

A *grand*, *unified* theory of cognition, at a minimum, would comprise theories of perception, the various kinds of memory, concepts, conceptual semantics, reference, emotion's role in cognition, as well as a theory of representation together with a theory of representational semantics, along with, of course, a specification of the functions actually responsible for cognition (one might rationally require theories of intentionality and consciousness to be added to the brew). This is an enormous list of needed theories, far larger than anywhere else in science. What are the dominant theories of, for example, perception, concepts, semantics, representations or

Journal of Experimental & Theoretical Artificial Intelligence
ISSN 0952–813X print/ISSN 1362–3079 online © 2003 Taylor & Francis Ltd
http://www.tandf.co.uk/journals DOI: 10.1080/0952813031000162408

intentionality that has guided cognitive science? There are none. We simply do not have anything close to a well-developed theory of these components of cognition. This lack of robust theories is frequently interpreted as a failure for cognitive science (it is not a failure, of course, cognition and associated other mental phenomena are among the most complicated in all of nature). This alleged failure is laid at the feet of computationalism. When 'theoreticians' (philosophers, mostly) wish to indict something for the 'failures' of cognitive science, they descend on computationalism precisely because it is the only component of a unified theory that is robust enough to attack. This explains the collection of essays in C*omputationalism: New Directions*.

2. What computationalism is and isn't

Computationalism is the *hypothesis* that cognition is best *explained* as algorithmic execution of Turing-computable functions. (Computationalism exclusively focuses on cognition. It does not say, at least at this stage, that emotions or consciousness or any of the other many mental phenomenon are Turing computable.) Such explanations have three properties: they explain cognitive *capacities*, they are *systematic* and they are *interpretive*.

- Computational explanations explain capacities or abilities of a system to exhibit certain behaviour. Hence, computational explanations differ from more well-known causal explanations in sciences such as physics, which subsume behaviour under a causal law.
- Computational explanations are systematic. For any system of reasonable complexity (say, anything from an arthropod on up), computational explanations posit interdependent functions that interact to produce the output from the input. Given a system, a computational explanation analyses the system-sized function—which explains the system's gross behaviour—into subfunctions and then shows how those subfunctions interact to produce the system's behaviour.
- Computational explanations are interpretive. In order to explain some bit of behaviour as computing function F, say, we must be able to interpret the initial state of the system as being the requisite input for F, the final state of the system ('final' in a teleological sense) as being the requisite output of F, and all of the system's intermediate states as being the subfunctions of F.

This last property has been misunderstood by some, even by computationalists (e.g. Chalmers, 1994). It is important to get the order of events right. We do not want to show how a computation is implemented in some system (contra Chalmers). Precisely the opposite: we want to show how some system implements a computation. This change of order is quite important. As cognitive scientists, we do not pick a computation, and then look for a system that implements it. Rather, we want to explain how some system does what it does by interpreting it as executing some computation. We begin with a physical system's behaviour and look for what explains it, where the explanans are computations performed over representations. To the computationalist, as with any scientist, it is a system's behaviour that is central. Interestingly, this is true even in artificial intelligence. True, in AI we do implement computer programs, but the point of the program is that it explains the behaviour of the machine, which in turn is used to model some aspect of intelligent behaviour. This point can be readily observed when one

considers scientific AI, rather than engineering AI (for two examples, see MAC/FAC (Forbus *et al.* 1995) and LISA (Hummel and Holyoak 1997). (Chalmers would probably agree that explanation grounds the use of computations, at least in part. But in his writings, this point gets lost. He seems more interested in the abstract computation and its implementation, rather than the physical system and its explanation, which goes precisely the opposite direction. We darkly suspect that Chalmers is more interested in metaphysics than is your typical cognitive scientist working in the field.)

The above is all the hypothesis says. It would seem to be hard to get it wrong. Yet, here are three common misunderstandings of computationalism. First, the computational hypothesis is only a foundational, abstract hypothesis (we will have more to say about this, below, when we discuss some essays from the book). All the hypothesis gives us is a framework within which to work. It does not get specific about which particular functions cognition is. Therefore, the hypothesis does not tell us what models to build, nor which experiments to run. *The computational hypothesis is not a theory of mind.*

Secondly, *the hypothesis is not committed to mental representations of any particular variety*. Rather, it is compatible with a myriad of different kinds of representations from mental spaces to numerical quantities to nodes in a semantic network. Together, these two points mean that the hypothesis leaves all the hard work to do. We still need to roll up our sleeves and get down to the difficult business of developing a theory of mind.

A third misunderstanding is to equate it with 'computerism'. Computerism is the view that the architectures that allow for cognition are structured like a von Neumann machine, i.e. a digital, serial computer. Computationalism is completely agnostic as to whether the specified functions, which are invoked to explain some systems behaviour, will be implemented on a classical von Neumann architecture or on a massively parallel distributed processor, a continuous processor or something that we have not imagined yet (it is extremely unlikely that von Neumann machines are up to the task). All that the hypothesis insists upon is that the specified functions be Turing-computable. (For more on computationalism, see Dietrich and Markman 2000 and Dietrich 1990a.)

Even though the hypothesis is broad and general, this in itself does not mean that it is vacuous or unfalsifiable. Much to the contrary: computationalism is a robust framework that could be unambiguously proven false. This could occur in at least two ways. First, a cognitive system could be shown to execute a non-Turing-computable function or, secondly, a cognitive system could be found to routinely and quickly solve arbitrarily large instances of the travelling salesman problem (i.e. NP-complete problems). Either option would be bad news for computationalism (the latter because it would mean that our understanding of the class NP would be lacking something crucial, and the class NP and theory of computation are intimately tied up with each other), but, until the day that such revelations are forthcoming, computationalism remains the best framework within which to do cognitive science.

3. Computationalism and its discontents

To keep things somewhat short, we review three essays (out of seven) from *Computationalism: New Directions*.

3.1. *Brian Cantwell Smith*

Brian Cantwell Smith does not attack computationalism head on; rather he changes the topic to a new understanding of computation! Smith's essay is entitled 'The Foundations of Computing'. According to Smith, all existing theories of computation are inadequate because they lack a proper theory of ontology on which to ground them (p. 48). Smith says that the theory of computation is not a theory of computation, when viewed properly (p. 42); rather it is 'neither more nor less than a *mathematical theory of the flow of causality*' (p. 43).

By requiring that any adequate theory of computation be a full-blown theory of causation complete with its own theory of ontology and semantics, it is clear that Smith completely shifts the goal posts on computationalists—clear out to the parking lot. We cannot stress enough how misguided this view is. A theory of computation *should not* be a full-blown theory everything, contra Smith (p. 53). To assert that it should completely misconstrues the place of computationalism as a foundational hypothesis in cognitive science. This is like indicting the theory of gravity because it fails to explain in detail why people build buildings or airplanes the way they do.

Nowhere is this more evident than when Smith holds that a criterion for any adequate theory of computation must be that the theory 'do justice to—by explaining or at least supplying the wherewithal with which to explain—the full range of computational practice' (p. 24). You read right: Smith is saying that any adequate theory of computation must supply a theory of how computers and other information processing devices are embedded in social practices. To conflate the theory of computation with computing practice (let alone their place in society) is to blur very important distinctions to point where all we have is mud.

Our interpretation of Smith might be a tad harsh, since at the end of his paper he drastically expands his focus to include not only computation, computers, and their social milieu, but also basically everything, including *all artefacts*! Concerning cognitive scientists, Smith writes that '[i]t is we, not the physicists, who must develop a theory of everything' (p. 53). So, at the end of the day, he may be doing (besides everything else), metaphysics. He certainly thinks he is doing metaphysics: 'the range of experience and skills and theories and results that have been developed within computer science—astoundingly complex and far-reaching, if still inadequately articulated—is best understood as practical, synthetic, raw material for no less than full theories of causation, semantics, and ontology—that is, for *metaphysics full bore*' (p. 52). If he is doing metaphysics, then he is after something other than what we are after: a well-behaved theoretical foundation for cognitive science. In such a case, it might make sense for him to look for what computation, computationalism and computing practice and society might have in common. Still, one can reasonably wonder why it is that Smith has so radically changed the rules of the game. And as matters now stand, it is a game no cognitive scientist should want to play.

(There is plenty of interesting philosophy to engage Smith in. As one example, why does he go to all this trouble, but still assume that there are *natural kinds*? Smith seems to buy into a classical kind of realism that assumes that the world comes with specific kinds of things in it (atoms, proteins, ducks, galaxies, etc.) and it is science that studies those things. He gets his radical conclusion by arguing that computation and computers are not natural kinds. But getting rid of the straitjacket assumption that there are natural kinds frees up the metaphysical and epistemological situation

considerably without having to completely recast the theory of computation as a theory of everything.)

3.2. *Jack Copeland*

At first glance, Copeland's essay, 'Narrow versus Wide Mechanism', seems to be an explication of notions of computation that extend beyond the Turing limit (i.e. devices that could compute functions that Universal Turing Machines could not, such as the halting problem). However, upon closer inspection, the real agenda is to make a case that human cognition is not adequately modelled by standard computation, i.e. that computationalism is false. Copeland presents his case by distinguishing between two types of mechanism. *Narrow mechanism* is 'the view that the mind is (strictly, can be simulated by) a Turing machine' (p. 63). Contrasted to this is *wide mechanism*, the view 'that the mind is a machine but [which view] countenances the possibility of information-processing machines that cannot be mimicked by a universal Turing machine, and allows in particular that the mind may be such a machine' (p. 63). From this distinction, Copeland argues: (1) that the notion of mechanism does not exhaust or even entail narrow mechanism; (2) that the Church-Turing Thesis favors neither the wide nor narrow views of mechanism; and (3) that asserting that all mechanism is exhausted by, or co-extensive with, narrow mechanism is to commit what Copeland calls the *Church-Turing Fallacy* (pp. 63–64, 72).

One thing that initially struck us about Copeland's essay is how much space Copeland devotes to Turing hermeneutics. Since this paper is not a historical paper, why all the exegesis? What we think is really going on here is an argument from authority. The argument seems to be that 'since even Turing had a notion of wide mechanism (i.e. of machines stronger than Universal Turing Machines), then there really is a solid notion of machines stronger than Universal Turing Machines—and minds are those things'.

The notion of a super-computation (or wide mechanism) is well established, so arguing that there is a coherent, mathematical notion of super-Turing machines by doing Turing exegesis is pointless. The crucial issue is not whether such a notion exists but what use such a notion has for cognitive science. The answer is that it has not any use at all. The reason it is not useful turns on an important epistemological quandary that Copeland simply misses. Super-computation (wide mechanism) is not useful because *we can never really know if we have successfully built or found such a machine*. Suppose, for the sake of argument, that one day we found a machine in our backyard labelled 'Super-Turing Machine'. We would have no way of determining that what was found in our backyard was in fact a Super-Turing machine. Every test we could ever run would give us results completely consistent with the view that what we had was a large Turing machine. Better yet, let's say God shows up one day and says, in a mellifluous voice, 'Here is a Super-Turing machine, trust me!' Such a proclamation if true would be mildly useful because we could get answers to lots of questions we could not have before. But, and this is the crucial point, we would have to *assume* we actually get the answers—i.e. we would literally have to trust Him blindly, for we could not prove that the machine was a Super-Turing Machine. At the end of the day, blind trust would not help that much for we would still be able quite easily and rationally to doubt that it was God that had given us the machine and not Satan, setting us up, as he is wont to do, for some disaster or other.

The crux of Copeland's essay is his alleged Church-Turing Fallacy. As Copeland defines it: 'to commit the Church-Turing fallacy is to believe that the Church-Turing thesis, or some formal or semiformal result established by Turing or Church, secures the proposition that, if mechanism is true, the functions generated by Turing machines provide sufficient mathematical resources for a complete account of human cognition' (p. 72). We suppose that this is a fallacy (a form of equivocation since 'mechanism' is ambiguous between narrow and wide mechanism), but we do not know anyone who commits it. Consider for example, Copeland's discussion of Fodor.

Copeland says that this sentence of Fodor's is false: 'If a mental process can be functionally defined as operation on symbols, there is a Turing machine capable of carrying out the computation' (p. 73). Copeland claims that this is false because oracle machines (Super-Turing machines) satisfy the antecedent, but the consequent is false. But it is quite clear that Fodor meant by the phrase 'operation on symbols' some effective, pencil-and-paper-only operation (the standard way 'algorithm' is defined; see Dietrich 1999). This definition of 'operation on symbols' together with the Church-Turing Thesis entails the consequent. Copeland is merely being uncharitable in his interpretation of Fodor. Furthermore, even if Fodor *et al.* had, through laziness, committed Copeland's alleged fallacy, so what? Ask them politely to stop. Require them to always put in the qualifier 'some effective, pencil-and-paper-only' in front of 'operation.' We are sure they would kindly oblige, for this is what they actually meant. (Note that adding the qualifier does not make Fodor's claim vacuous, since you really do need the Church-Turing Thesis to get the consequent.)

In the final pages of Copeland's essay, he writes that 'the crucial issue here is whether our cognitive architecture, abstracted from resource constraints, is best understood as being a generator of (one or more) Turing-machine-uncomputable functions, and the fact that the mind is simulable by Turing machine when certain resource constraints are operative says nothing either way' (p. 83). We cannot overstress how wrong-headed this is. This is *not* the crucial issue, for we live in a universe of constraints. Knowing that we are Super-Turing machines *sans* constraints brings us no comfort at all—there would be peace in the Middle East if the Israelis and the Palestinians could learn to get along, or we would all be immortal if all the constraints that conspired in our death were removed—who would doubt it? The fact of the matter is that we live in a world of constraints—constraints that make life possible in the first place. What we really want to know is what a mind and its brain are in this universe, not in some non-universe of no constraints. Of what use would knowing that be? (There is an interesting and deep technical point here. With our constraints in place, we are in fact finite state machines—far simpler than Turing machines. But, *pragmatically*, Turing machine functions are the notion to use because they in fact allow us the flexibility to theorized about human creativity and reasoning power in way that using FSMs cannot.)

3.3. *Aaron Sloman*

Finally, Aaron Sloman's essay, 'The Irrelevance of Turing Machines to Artificial Intelligence', pointedly claims just what the title says: that Turing machines are irrelevant to AI. We agree with a lot of what Sloman says (see Dietrich 1990b). As he correctly points out, 'the development of computers owed nothing to the idea of a Turing machine or the mathematical theory of computation' (p. 101). And we

agree with his positive statement: 'computers . . . are, above all, engines for acquiring, storing, analysing, transforming, and using information' (p. 125). The question is what inference should be drawn from these facts.

It is clear that quotidian AI (as well as quotidian psychology, linguistics and anthropology) does not need Turing machines or the theory of computation. But this, in and of itself, is not sufficient to establish its irrelevance to cognitive science or AI. The reason is that modern cognitive science rests on the computational theory of mind. In day-do-day jargon, this can couched as the claim that minds are engines for 'acquiring, storing, analysing, transforming and using information just as are those boxes over there (pointing to, say, your beloved Powerbook)'. That is a fine way to put things, but it is also appropriate and right to get more abstract and global, and ask 'What kind of machine is an engine for acquiring, storing, analysing, transforming and using information?' The answer will turn out to be: a Turing machine.

We are claiming that, correctly understood, Turing machines and the attendant theory of computation, are relevant to cognitive science and AI because *they provide a high-level explanation of what the field is about*. Understood in this way, the philosophical version of the computational theory of mind is really *philosophy of science*: it is an abstract description of cognitive science's global trajectory. This view of the relationship between Turing machine theory and cognitive science explains how and where Sloman's view is correct: Turing machines are irrelevant to AI and cognitive science because there is no causal or informational relation *from* Turing machine theory *to* cognitive science. Rather, the flow of information is the other way: from cognitive science and AI to the Turing machine view of these disciplines. This is how Turing machine theory acquires its relevance to AI and cognitive science. If Sloman is right, the day-to-day life of cognitive scientists will continue as it always has—they will propose algorithms for explaining cognitive capacities. But one good consequence of our view is that such behaviour on the scientists part will continue to be well-explained by computationalism.

4. Conclusions

Scheutz begins his introductory essay with an extended analogy between *Star Trek* and *Star Trek: The Next Generation*. His book is supposed to be the latter, while older versions of computationalism (whatever those are) are the former. Both versions of *Star Trek* are really retreaded 19th-century British empire sailing stories: go out, sail the world, meet strange new people, have adventures. This really *is* science fiction. Nothing even remotely like that is going to happen. But what the heck, it's just a fun metaphor. It is clear, though, that the better metaphor is *Star Wars*; *Computationalism: New Directions* is really new flirtations with the Dark Side.

References

Chalmers, D., 1994, A computational foundation for cognition. *Minds and Machines*, **5**(4).

Dietrich, E., 1990a, Computationalism. *Social Epistemology*, **4**(2), 135–154 (with commentary). Reprinted in E. Dietrich (eds), 1994, *Thinking Computers and Virtual Persons: Essays on the Intentionality of Machines* (San Diego: Academic Press).

Dietrich, E., 1990b, Programs in the search for intelligent machines: the mistaken foundation of AI. In D. Partridge and Y. Wilks (eds), *The Foundations of Artificial Intelligence: A Source Book* (Cambridge: Cambridge University Press).

Dietrich, E., 1999, Algorithm. Entry in R. Wilson and F. Keil (eds), *MIT Encyclopedia of the Cognitive Sciences* (Boston, MA: MIT Press), pp. 11–12.

Dietrich, E. and Markman, A., 2000, Cognitive dynamics: computation and representation regained. In E. Dietrich and A. Markman (eds), *Cognitive Dynamics: Conceptual change in humans and machines* (Mahwah, NJ: Lawrence Erlbaum).

Forbus, K., Gentner, D. and Law, K., 1995, MAC/FAC: a model of similarity-based retrieval. *Cognitive Science*, **19**, 141–205.

Hummel, J. and Holyoak, K. J., 1997, Distributed representations of structure: a theory of analogical mapping. *Psychological Review*, **104**(3), 427–466.